TUGAS AKHIR – TF 145565

SISTEM MONITORING *FLOW* DAN *LEVEL* PADA RANCANG BANGUN PEMBANGKIT LISTRIK TENAGA MIKRO HIDRO (PLTMH)

RAGIL SURYANING FAJAR
NRP. 10 51 15 00000 015

Dosen Pembimbing I
Detak Yan Pratama, S.T., M.Sc.
NIP. 19840101 201212 1 002

Dosen Pembimbing II
Sefi Novendra Patrialova, S.Si., M.T.
NPP. 1991 20171 2 053

PROGRAM STUDI DIII TEKNOLOGI INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya
2018

TUGAS AKHIR – TF 145565

SISTEM MONITORING *FLOW* DAN *LEVEL* PADA RANCANG BANGUN PEMBANGKIT LISTRIK TENAGA MIKRO HIDRO (PLTMH)

RAGIL SURYANING FAJAR
NRP. 10 51 15 00000 015

Dosen Pembimbing I
Detak Yan Pratama, S.T., M.Sc.
NIP. 19840101 201212 1 002

Dosen Pembimbing II
Sefi Novendra Patrialova, S.Si., M.T.
NPP. 1991201712053

PROGRAM STUDI DIII TEKNOLOGI INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya
2018

LEMBAR PENGESAHAN I

SISTEM MONITORING FLOW DAN LEVEL PADA RANCANG BANGUN PEMBANGKIT LISTRIK TENAGA MIKRO HIDRO (PLMTH)

TUGAS AKHIR

Oleh:

Ragil Suryaning Fajar
NRP. 105 115 00000 015

Surabaya, 25 Juli 2018
Mengetahui dan Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Detak Yan Pratama. ST.. M.Sc.	**Sefi Novendra Patrialova. S.Si.. MT.**
NIP. 19840101 201212 1 002	**NPP. 1991201712053**

Kepala Departemen Teknik Instrumentasi

Dr.Ir.Purwadi Agus Darwito. M.Sc.
NIP. 19620822 198803 1 001

LEMBAR PENGESAHAN II

SISTEM MONITORING FLOW DAN LEVEL PADA RANCANG BANGUN PEMBANGKIT LISTRIK TENAGA MIKRO HIDRO (PLMTH)

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat Memperoleh Gelar
Ahli Madya
Pada Program Studi DIII Teknologi Instrumentasi
Departemen Teknik Instrumentasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Oleh :

Ragil Suryaning Fajar
NRP. 105 115 00000 015

Disetujui oleh Tim Penguji Tugas Akhir :

1. Detak Yan Pratama, ST., M.Sc.....(Dosen Pembimbing I)
2. Sefi Novendra Patrialova, S.Si., MT.....(Dosen Pembimbing II)
3. Ahmad Fauzan Adziima S.T, M.Sc.....(Dosen Penguji)

SURABAYA
JULI 2018

SISTEM MONITORING FLOW DAN LEVEL PADA RANCANG BANGUN PEMBANGKIT LISTRIK TENAGA MIKRO HIDRO (PLMTH)

Nama : Ragil Suryaning Fajar
NRP : 10511500000015
Departemen : Teknik Instrumentasi FV-ITS
Pembimbing 1 : Detak Yan Pratama, S.T., M.Sc.
NIP. 19840101 201212 1 002
Pembimbing 2 : Sefi Novendra Patrialova, S.Si, M.T.
NPP. 1991 20171 2 053

Abstrak

Pembangkit Listrik Tenaga Mikro Hidro (PLTMH) merupakan pembangkit listrik yang memanfaatkan tenaga (aliran) air sebagai sumber penghasil energi. Karena aliran air adalah sumber utama, maka diperlukan sistem monitoring pada *flow* dan *level* untuk memantau proses mengalirnya air yang akan jatuh ke turbin. Sensor yang digunakan untuk memantau *level* adalah sensor ultrasonik SRF05 dan untuk memantau *flow* adalah sensor *waterflow* FS400a dengan mikrokontroller ATmega128. Hasil pembacaan kedua sensor tersebut dapat ditampilkan melalui LCD 20x4 yang terpasang pada *panel box* dan juga dapat ditampilkan melalui HMI (*Human Machine Interface*) yang ada pada PC (*Portable Computer*). Setiap kali proses dijalankan, maka data juga akan tersimpan pada *data logger* yang menggunakan modul *open log* dengan SD *Card* sebesar 4gb. Pembacaan sensor pada variabel *level* memiliki tingkat akurasi sebesar 99,026% dan pada variabel *flow* sebesar 98,355%.

Kata Kunci: Sistem *Monitoring Flow*, Sistem *Monitoring Level*, HMI, *Data Logger*.

MONITORING SYSTEM OF FLOW AND LEVEL IN THE DESIGN OF THE MICRO HYDRO POWER PLANT (PLTMH)

Name	***: Ragil Suryaning Fajar***
NRP	***: 10511500000015***
Department	***: Instrumentation Engineering FV-ITS***
Advisor 1	***: Detak Yan Pratama, S.T., M.Sc***
	NIP. 19840101 201212 1 002
Advisor 2	***: Sefi Novendra Patrialova, S.Si, M.T.***
	NPP. 1991 20171 2 053

Abstract

Micro Hydro Power Plant (PLTMH) is a power plant that utilizes the power (stream) of water as a source of energy. Because the water flow is the main source, a monitoring system on the flow and level is needed to monitor the flow of water that will fall into the turbine. The sensor used to monitor the level is the ultrasonic sensor SRF05 and to monitor flow is the FS400a waterflow sensor with ATmega128 microcontroller. The reading results of both sensors can be displayed through the LCD 20x4 mounted on the panel box and can also be displayed through the HMI (Human Machine Interface) on the PC (Portable Computer). Every time the process is run, the data will also be stored in a data logger that uses an open log module with an SD Card of 4GB. Sensor reading on the level variable has an accuracy level of 99.026% and the variable flow is 98.355%.

Keywords*: Flow Monitoring System, Level Monitoring System, HMI, Data Logger*

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas berkat, rahmat dan kebesaran-Nya sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul **"Sistem Monitoring Flow Dan Level Pada Rancang Bangun Pembangkit Listrik Tenaga Mikro Hidro (PLTMH)"** tepat pada waktunya.

Selama menyelesaikan tugas akhir ini penulis telah banyak mendapatkan bantuan dari berbagai pihak. Oleh karena itu penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Keluarga tercinta yang telah memberikan segala semangat, do'a, dan dukungan baik moral maupun materil yang sangat luar biasa.
2. Bapak Dr. Ir. Purwadi Agus Darwito, M.Sc selaku Kepala Departemen Teknik Instrumentasi ITS yang telah meluangkan waktu selama proses pengerjaan tugas akhir.
3. Bapak Ir. Ya'umar selaku dosen wali yang telah memberikan semangat dan motivasi.
4. Bapak Detak Yan Pratama, S.T., M.Sc., selaku dosen Pembimbing 1 yang telah meluangkan waktu, memberi saran dan arahan selama proses pengerjaan tugas akhir.
5. Ibu Sefi Novendra Patrialova, S.Si., M.T., selaku dosen pembingbing 2 yang telah meluangkan waktu, memberi saran dan arahan selama proses pengerjaan tugas akhir.
6. Seluruh civitas akademika Departemen Teknik Instrumentasi yang telah banyak membantu selama ini.
7. Kelompok PLTMH (Nila Diah, Delima Palwa, M. Arifur, Hanif Adi, Ibrahim Agam) yang telah sama-sama berjuang dalam pengerjaan tugas akhir ini.
8. Teman - teman F50, khususnya F50.3 yang telah mendukung, mambantu, dan menemani selama kuliah hingga terselesainya tugas akhir ini.
9. Teman – teman himpunan, bem fakultas, bem ITS, dan gerigi yang senantiasa memberikan semangat dan dukungan.
10. Serta semua pihak yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa masih terdapat kekurangan dalam laporan kerja praktek ini, maka dari itu penulis mengharap kritik dan saran yang membangun agar penyusunan laporan selanjutnya lebih baik lagi. Semoga laporan kerja praktek ini dapat memberikan manfaat bagi kita semua.

Surabaya, Juli 2018
Penulis

Ragil Suryaning F.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Salah satu potensi sumber daya alam terbesar yang dimiliki oleh Bangsa Indonesia adalah air. Di samping kegunaannya untuk memenuhi kebutuhan hidup sehari –hari, kandungan energi yang dimiliki oleh air yang mengalir dari ketinggian tertentu dan jumlah tertentu juga bisa dimanfaatkan sebagai pembangkit energi mekanis. Salah satu contoh alat konversi energi air menjadi energi mekanik adalah turbin air, energi mekanik pada tubin air dapat di ubah menjadi energi listrik yang merupakan salah satu sumber energi alternative yang dapat diperbaharui. Dalam memproduksi energi listrik yang bisa diperbarui memanfaatkan sumber tenaga air dalam skala kecil yang dikenal juga dengan Pembangkit Listrik Tenaga Mikro Hidro (PLTMH). Data hasil survey potensi tenaga air yang dilakukan PLN pada tahun 1982, diseluruh Indonesia terdapat potensi untuk pengembangan PLTA dan PLTMH diperkirakan sebesar 75.000 MW. Diantara potensi tersebut terdapat potensi tenaga air untuk mikro hidro. Menurut Rencana Induk Pengembangan Energi Baru dan Terbarukan (RIPEBAT), potensi tenaga air mikro hydro saja diperkirakan 458,75 MW (Chayun Budiono , 2003).

Tenaga mikrohidro, dengan skala daya yang dapat dibangkitkan 5 kilo watt hingga 50 kilo watt. Pada PLTMH proses perubahan energi kinetik berupa (kecepatan dan tekanan air), yang digunakan untuk menggerakan turbin air dan generator listrik hingga menghasilkan energi listrik (NOTOSUDJONO, D. 2002).

Pembangkit yang memanfaatkan kondisi alam, menyebabkan energi listrik yang dihasilkan sangat fluktuatif tergantung pada kondisi cuaca. Arus dan tegangan pada jaringan biasanya kurang stabil, maka dibutuhkan sistem monitoring pada jaringan untuk memantau besarnya arus dan tegangan. (Fitriandi, Komalasari, & Gusmedi, Rancang Bangun Alat Monitoring Arus dan Tegangan Berbasis, 2016)

Pada jurnal tersebut, pemantauan hanya dilakukan pada arus dan tegangan. Oleh karena itu, penulis membuat tugas akhir

berjudul "Sistem *Monitoring Flow* dan *Level* Pada Rancang Bangun Pembangkit Listrik Tenaga Mikro Hidro (PLTMH)" untuk dapat diketahui besarnya ketinggian air pada tangki dan besarnya debit air yang mengalir sebelum jatuh ke turbin pada PLTMH.

1.2 Rumusan Masalah

Adapun masalah yang dapat dirumuskan dari tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara untuk merancang sistem *monitoring flow* dan *level* pada Pembangkit Listrik Tenaga Mikro Hidro (PLTMH),
2. Bagaimana cara untuk menyimpan dan menampilkan hasil data pembacaan dari suatu *flow* dan level tangki pada Pembangkit Listrik Tenaga Mikro Hidro (PLTMH).

1.3 Tujuan

Adapun tujuan dari tugas akhir ini adalah sebagai berikut.

1. Merancang sistem *monitoring flow* dan *level* pada Pembangkit Listrik Tenaga Mikro Hidro (PLTMH).
2. Menyimpan dan menampilkan hasil data pembacaan dari suatu *flow* dan level tangki pada Pembangkit Listrik Tenaga Mikro Hidro (PLTMH).

1.4 Batasan Masalah

Adapun batasan masalah yang ada pada Tugas Akhir ini adalah sebagai berikut.

1. Variabel yang di*monitoring* adalah *flow* dan *level*.
2. Mikrokontroleryang digunakan adalah ATmega128.
3. Sensor yang digunakan adalah sensor ultrasonik (untuk level) dan sensor *water flow* (untuk *flow*).
4. Penyimpanan data menggunakan *data logger* dengan modul *open log*.
5. HMI (*Human Machine Interface*) hanya melakukan *monitoring* tanpa adanya pengendalian atau kontrol.
6. Bahasa pemrograman HMI yang digunakan adalah .vb (visual basic) pada Visual Studio 2015.

1.5 Manfaat

Manfaat yang didapatkan dari tugas akhir ini adalah sebagai berikut.

1. Untuk departemen, tugas akhir ini dapat dijadikan salah satu alat untuk praktikum.
2. Untuk mahasiswa, dapat digunakan sebagai pembelajaran dan referensi untuk pengembangan alat yang lebih baik.

"Halaman ini sengaja dikosongkan"

BAB II
DASAR TEORI

2.1 Pembangkit Listrik Tenaga Mikro Hidro (PLTMH)

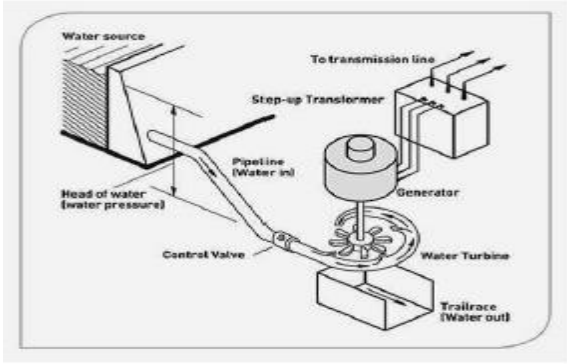
Pembangkit Listrik Tenaga Mikrohidro (PLTMH) adalah pembangkit listrik berskala kecil (kurang dari 100 kW), yang memanfaatkan tenaga (aliran) air sebagai sumber penghasil energi. PLTMH termasuk sumber energi terbarukan dan layak disebut *clean energy* karena ramah lingkungan. Tenaga air berasal dari aliran sungai kecil atau danau yang dibendung dan kemudian dari ketinggian tertentu dan memiliki debit yang sesuai akan menggerakkan turbin yang dihubungkan dengan generator listrik. Semakin tinggi jatuhan air maka semakin besar energi potensial air yang dapat diubah menjadi energi listrik. Bentuk pembangkit tenaga mikro hidro bervariasi tetapi prinsip kerjanya adalah sama, yaitu: "Perubahan tenaga potensial menjadi tenaga elektrik (listrik)". Perubahan memang tidak langsung, tetapi berturut-turut melalui perubahan sebagai berikut:

Gambar 2.1 Perubahan Tenaga Potensial Menjadi Listrik
(Sumber: dokumen pribadi)

Tenaga potensial adalah tenaga air karena berada pada ketinggian. Energi kinetik adalah tenaga air karena mempunyai kecepatan. Tenaga mekanik adalah tenaga kecepatan air yang terus memutar kincir/turbin. Tenaga listrik adalah hasil dari generator yang berputar akibat berputarnya kincir/turbin. (Prayogo, 2003)

Pembangkit listrik tenaga air skala mikro pada prinsipnya memanfaatkan beda ketinggian dan jumlah debit air per detik yang ada pada aliran air saluran irigasi, sungai atau air terjun. Aliran air ini akan memutar poros turbin sehingga menghasilkan energi mekanik. Energi ini selanjutnya menggerakkan generator dan generator menghasilkan listrik. Sebuah skema mikrohidro

memerlukan dua hal yaitu, debit air dan ketinggian jatuh (*head*) untuk menghasilkan tenaga yang dapat dimanfaatkan. Hal ini adalah sebuah sistem konversi energi dari bentuk ketinggian dan aliran (energi potensial) kedalam bentuk energi mekanik dan energi listrik. (Fox & Mc. Donald, 1994)



Gambar 2.2 Sistem PLTMH
(Source: (Sihaloho, 2017))

Cara kerjanya adalah aliran sungai dibendung agar mendapatkan debit air (Q) dan tinggi jatuh air (H), kemudian air yang dihasilkan disalurkan melalui saluran penghantar air menuju kolam penenang, Kolam penenang dihubungkan dengan pipa pesat, dan pada bagian paling bawah di pasang turbin air. Turbin air akan berputar setelah mendapat tekanan air (P), dan perputaran turbin dimanfaatkan untuk memutar generator, Setelah mendapat putaran yang constan maka generator akan menghasilkan tegangan listrik, yang dikirim ke konsumen melalui saluran kabel distribusi (JTM atau JTR). (Sukamta & Kusmantoro, 2013)

2.2 Sensor *Level* dan Sensor *Flow* Pada PLTMH

Sensor yang terdapat pada rancang bangun PLTMH dari saat air ditampung pada tangki hingga saat sebelum air jatuh mengenai turbin ini, ada 2 buah. Yaitu sensor *flow* yang menggunakan sensor *water flow* dan sensor *level* yang menggunakan sensor ultrasonik.

2.2.1 Sensor *Water Flow*

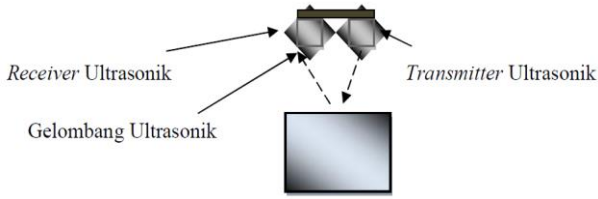
Sensor *Water Flow* terdiri dari bodi katup plastik, rotor air dan sensor *hall effect*. Ketika air mengalir melalui rotor, maka rotor akan berputar sesuai dengan kecepatan aliran air yang mengalir melalui rotor tersebut. Prinsip kerja sensor ini adalah dengan memanfaatkan sensor *hall effect*. *Hall effec*t ini didasarkan pada efek medan megnetik terhadap partikel bermuatan yang bergerak. Ketika ada arus listrik yang mengalir pada *hall effect* yang ditempatkan dalam medan magnet yang arahnya tegak lurus arus listrik, pergerakan pembawa muatan akan berbelok ke salah satu sisi dan menghasilkan medan listrik. Medan listrik terus membesar hingga gaya Lorentz yang bekerja pada partikel menjadi nol. Perbedaan potensial antara kedua sisi device tersebut disebut potensial *Hall*. Potensial *Hall* ini sebanding dengan medan magnet dan arus listrik yang melalui *device*. (Purnama, 2012)

Gambar 2.3 Sensor *Water Flow* Diameter 1”
(Sumber: www.hotmcu.com/g1-water-flow-sensor-p-312.html?cPath=8)

2.2.2 Sensor Ultrasonik

Metode yang digunakan untuk mengukur tinggi muka air dengan ultrasonik menggunakan prinsip *echosounder*. Sesuai dengan namanya dalam bahasa Inggris, *echo* berarti gema dan *sounder* berarti pemancar bunyi, maka metode ini memanfaatkan pemancaran pulsa ultrasonik dari transmitter ultrasonik dengan frekuensi sebesar 40 kHz dan kemudian gemanya atau pemantulannya yang timbul akibat mengenai suatu benda yakni

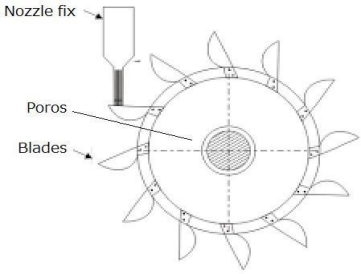
muka air akan diterima kembali oleh receiver ultrasonik. (Nadiya, 2016)



Gambar 2.4 Metode *Enchosounder* Untuk Mengukur Tinggi Muka Air
(Sumber: (Nadiya, 2016))

Gelombang ultrasonik ini merambat melalui udara dengan 344 m/s. Jarak minimal antara transmitter-receiver dengan permukaan air adalah 2 cm dengan jarak maksimal 300 cm. Setelah gelombang pantulan terdektesi akan dibuat output tertentu sebagai tanda bahwa gelombang sudah diterima untuk mematikan timer pengukur 15 waktu pulsa dari transmitter sampai receiver. Dengan mengukur selang waktu antara saat pulsa dikirim dan pulsa diterima, jarak antara muka air dengan ultrasonik dapat dihitung dengan persamaan s = v.t / 2. S adalah jarak antara muka air dengan sensor (m) dan v adalah laju bunyi (m/s). Jarak antara muka air dan sensor ultrasonik inilah yang dianggap tinggi muka air. (Nadiya, 2016)

2.3 Turbin Air Sebagai Penggerak Generator



Gambar 2.5 Sketsa Kasar Turbin Air
(Sumber: (Apriansyah, Rusdinar, & Darlis, 2016))

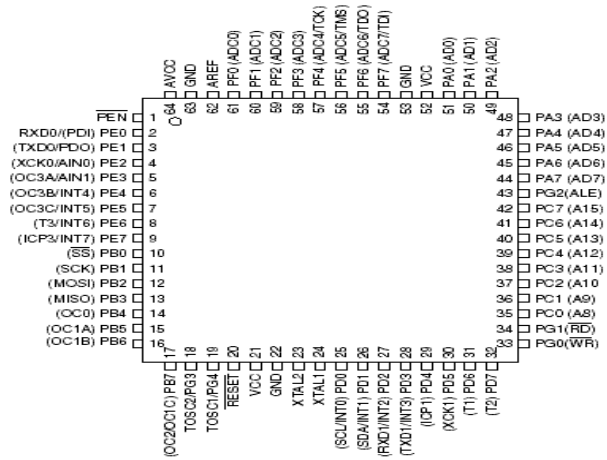
Turbin secara umum dapat diartikan sebagai mesin penggerak mula di mana energi fluida kerja yang digunakan langsung memutar roda turbin, fluida kerjanya dapat berupa air, dapat diartikan sebagai suatu mesin penggerak mula yang fluida kerjanya adalah air. Kalau ditinjau dari daya yang dihasilkan turbin air, maka dikenal istilah Pembangkit Listrik Tenaga Mini Hidro (PLTM) yang maksudnya adalah turbin air yang dapat menghasilkan daya kurang dari 5 MW dan sumber airnya relatif kecil. (Arismunandar, 1997)

Pemilihan jenis turbin dapat ditentukan berdasarkan kelebihan dan kekurangan dari jenisjenis turbin, khususnya untuk suatu desain yang sangat spesifik. Faktor tinggi jatuhan air efektif (*Net Head*) dan debit yang akan dimanfaatkan untuk operasi turbin merupakan faktor utama yang mempengaruhi pemilihan jenis turbin. (Ismono, 1999)

2.4 Sistem Monitoring

Dalam sistem monitoring, salah satu komponen penting yang digunakan adalah Atmega. Atmega yang dipakai pada tugas akhir ini adalah Atmega128.

AVR merupakan seri mikrokontroler CMOS 8-bit buatan Atmel, berbasis arsitektur RISC (Reduced Instruction Set Computer). Hampir semua instruksi dieksekusi dalam satu siklus clock. AVR mempunyai 32 register generalpurpose, timer/counter fleksibel dengan mode compare, interrupt internal dan eksternal, serial UART, programmable Watchdog Timer, dan mode power saving. Mempunyai ADC dan PWM internal. AVR juga mempunyai In-System Programmable Flash on-chip yang mengijinkan memori program untuk diprogram ulang dalam sistem menggunakan hubungan serial SPI. Salah satu contoh Atmega yang memiliki konfigurasi pin yang banyak adalah Atmega128. Atmega128 adalah mikrokontroler CMOS 8-bit daya-rendah berbasis arsitektur RISC yang ditingkatkan. Kebanyakan instruksi dikerjakan pada satu siklus clock, Atmega128 mempunyai throughput mendekati 1 MIPS per MHz membuat disainer sistem untuk mengoptimasi komsumsi daya versus kecepatan proses. (Arifin & Fathoni, 2014)



Gambar 2.6 Konfigurasi Pin ATmega 128
(Sumber: (Arifin & Fathoni, 2014))

Untuk pemrograman yang dapat digunakan untuk mengisi program pada mikrokontroller AVR, salah satunya adalah CodeVision-AVR. CodeVision-AVR adalah sebuah *compiler* C yang telah dilengkapi dengan fasilitas *Integrated Development Environment* (IDE) dan didesain agar dapat menghasilkan kode program secara otomatis untuk mikrokontroler Atmel AVR. Untuk mengingkatkan kehandalan program ini, maka pada CodeVisionAVR juga terdapat kumpulan pustaka (library) untuk:

- Modul LCD Alphanumeric
- Philips I2C bus
- National Semiconductor Sensor Temperatur LM75
- Philips PCF8563, PCF8583, dan Maxim/Dallas Semiconductor Real Time Clock DS1302 dan DS1307
- Maxim/Dallas Semiconductor 1 wire protocol
- Maxim/Dallas Semiconductor Sensor Temperatur DS1820, DS18S20, dan DS18B20

- Maxim/Dallas Semiconductor Termometer/Thermostat DS1621
- Maxim/Dallas Semiconductor EEPROMs DS2430 dan DS2433
- SPI
- Power Management
- Delays
- Gray Code Conversion
- MMC/SD/SD HC Flash memory cards low level access
- Akses FAT pada MMC/SD/SD HC Flash memory card

CodeVisionAVR dapat menghasilkan kode program secara otomatis melalui fasilitas CodeWizardAVR Automatic Program Generator. Dengan adanya fasilitas ini maka penulisan program dapat dilakukan dengan cepat dan lebih efisien.

2.5 Data Logger

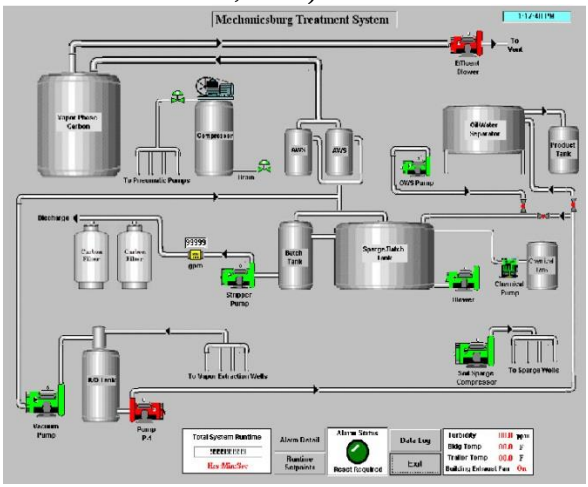
Merupakan suatu alat yang berfungsi untuk melakukan perekaman data, yaitu perangkat elektronika yang digunakan untuk mencatat data dari rentang waktu tertentu, biasanya prangkat ini bekerja bersama dengan sensor tertentu sesuai dengan kebutuhan. Proses pengumpulan dan perekaman data yang bejalan secara otomatis disebut data logging. Data logger dapat disimpan secara otomatis dengan format penyimpanan yang berbeda-beda, contoh penyimpanan data logger ditunjukan pada gambar berikut. (Fitriandi, Komalasari, & Gusmedi, Rancang Bangun Alat Monitoring Arus dan Tegangan Berbasis Mikrokontroler dengan SMS Gateway, 2016)

No	Tanggal	Waktu	Tegangan (Volt)	Arus (A)
1	4/10/2015	9:29:56	214.05	0.64
2	4/10/2015	9:34:50	215.78	0.28
3	4/10/2015	9:39:55	214.05	0.13
4	4/10/2015	9:44:50	212.67	0.15
5	4/10/2015	9:49:56	215.43	0.1
6	4/10/2015	9:54:50	211.97	0.12
7	4/10/2015	10:04:50	215.09	0.07

Gambar 2.7 Contoh *Data Logger*
(Sumber: Fitriandi, Komalasari & Gusmedi, 2016)

2.6 *Human Machine Interface* (HMI)

Human Machine Interface (HMI) adalah sistem yang menghubungkan antara manusia dan mesin. HMI dapat berupa pengendali dan visualisasi status, baik dengan manual maupun melalui visualisasi komputer yang bersifat real time. Tujuan digunakan-nya HMI adalah untuk meningkatkan interaksi antara operator dan mesin melalui tampilan di layar monitor. HMI mempunyai kemampuan dalam hal visualisasi untuk monitoring dan data mesin yang terhubung secara online dan real time. (CV. BINTANG JAYA TEKNIK, 2012)



Gambar 2.8 Contoh Aplikasi HMI di Mesin Industri
(Sumber: bintangjayateknik.com)

HMI akan memberikan suatu gambaran kondisi mesin yang berupa peta mesin produksi di layar monitor dimana dapat dilihat bagian mesin mana yang sedang bekerja. Sistem HMI biasanya bekerja secara online dan real time dengan membaca data yang dikirimkan melalui I/O port yang digunakan oleh controller. Port yang biasanya digunakan untuk controller dan akan dibaca oleh HMI antara lain adalah port com, port USB, port RS232 dan ada pula yang menggunakan port serial. HMI terdapat juga visualisasi pengendali mesin berupa push button, input reference dan sebagainya yang dapat difungsikan untuk mengontrol atau mengendalikan mesin sebagaimana mestinya. Selain itu pada HMI dapat ditampilkan alarm jika terjadi kondisi bahaya di dalam mesin. Sebagai tambahan, HMI dapat juga menampilkan data-data rangkuman kerja mesin secara grafik. (CV. BINTANG JAYA TEKNIK, 2012)

2.7 Karakteristik Statik

Karakteristik statik adalah sifat sebuah instrumen yang tidak tegantung pada waktu. Karakteristik statik merupakan hubungan yang terjadi antara output O dan input I dari sebuah elemen ketika I bernilai konstan maupun berubah perlahan. Adapun yang termasuk dalam bagian dari karakteristik static adalah sebagai berikut. (Karakteristik Statik Elemen Sistem Pengukuran)

a. *Range* (jangkauan)

Range (jangkauan) menyatakan jangkauan pengukuran sebuah instrumen.

b. *Span*

Span adalah variasi maksimum pada input atau output, yaitu $I_{\max} - I_{\min}$ atau $O_{max} - O_{min}$

c. Linieritas

Pengukuran yang ideal adalah jika hubungan antara input pengukuran (nilai sesungguhnya) memberikan *output* (nilai yang ditunjukkan alat ukur) yang berbanding lurus. Sebuah elemen dikatakan linier jikai nilai I dan O terletak pada

sebuah garis lurus, yang dinyatakan dalam persamaan berikut.

$$O_{ideal} = KI + \alpha \quad (2.1)$$

Dengan K adalah kemiringan garis:

$$K = \frac{Omax - Omin}{Imax - Imin} \quad (2.2)$$

Dan α adalah pembuat nol (*zero bias*):

$$\alpha = O_{min} - KI_{min} \quad (2.3)$$

$$\hat{N} = \frac{[O - KI + \alpha]max}{Omax - Omin} 100\% \quad (2.4)$$

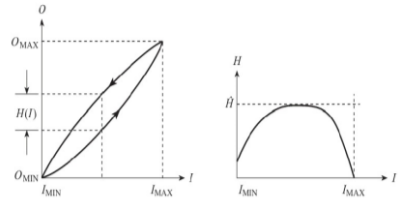
d. Sensitivitas

Karakteristik ini menunjukkan seberapa jauh kepekaan sensor terhadap kuantitas yang diukur. Secara matematis, sensitivitas menyatakan rasio *ΔO/ΔI*. Untuk elemen linier, sensitivitas adalah sama dengan kemiringan atau grdien garis K.

e. Histerisis

Histerisis menunjukkan perbedaan antra nilai output pembacaan pada saat menggunakan nilai input naik dengan nilai output pembacaan saat menggunakan nilai input turun. Histerisis biasanya dinyatakan dengan persamaan berikut.

$$\hat{\hat{H}} = \frac{O\downarrow - O\uparrow}{Omax - Omin} 100\% \quad (2.5)$$



Gambar 2.9 Histerisis
(Sumber: Karakteristik Statik Elemen Sistem Pengukuran)

f. Efek Lingkungan

Output dari sebuah elemen sistem pengukuran dipengaruhi oleh innput yang masuk ke elemen tersebut. Selain input dari besaran yang diukur, input lingkungan juga akan mempengaruhi output elemen. Seperti temperatur, tekanan atmosfer, kelembaban, tegangan, dan sebagainya. Akibat dari efek lingkungan tersebut, persamaan output elemen pengukuran menjadi:

$$O = KI + a + N(I) + K_M I_M I + K_I I_I \qquad (2.6)$$

g. Akurasi

Ketepatan alat ukur dalam memberikan hasil pengukuran.

$$A = 1 - \left|\frac{Yn - Xn}{Xn}\right| \times 100\% \qquad (2.7)$$

$$\text{Persen A} = A \times 100\% \qquad (2.8)$$

Dimana:

Y_n = Data yang standar (sesungguhnya)

X_n = Data yang terukur

h. Error

Error menunjukkan seberapa besar ketidak tepatan atau kesalahan dari hasil pembacaan sensor.

$$e = 1 - A \qquad (2.9)$$

$$\text{Persen e} = e \times 100\% \qquad (2.10)$$

i. Presisi

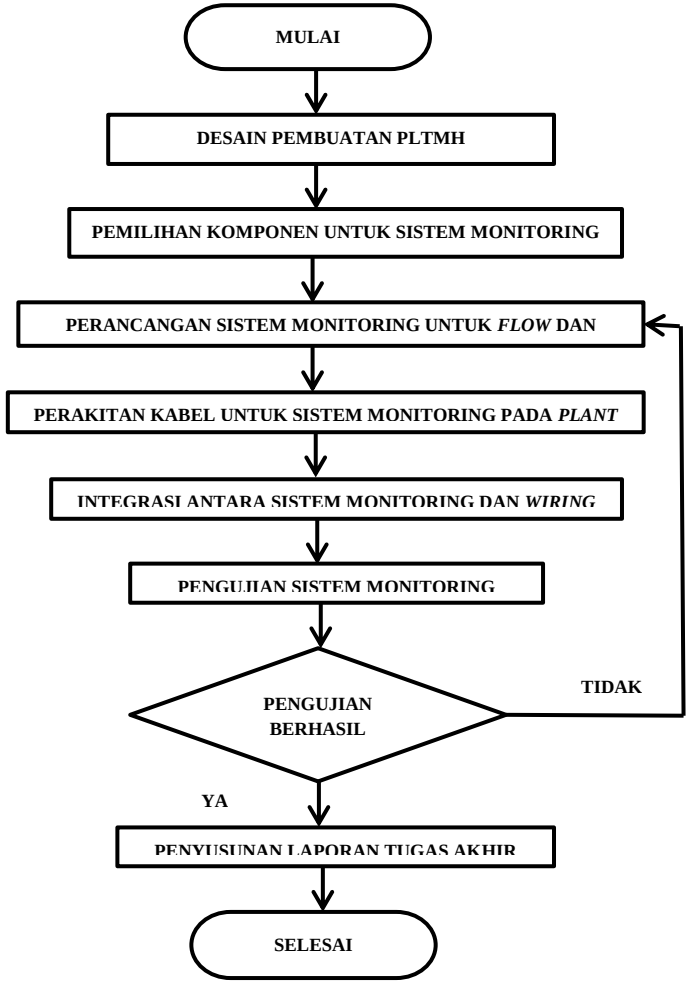
Presisi menyatakan derajat kebebasan sebuah instrumen dari kesalahan acak.

“Halaman ini sengaja dikosongkan”

BAB III
METODOLOGI

3.1 Prosedur Pengerjaan Alat

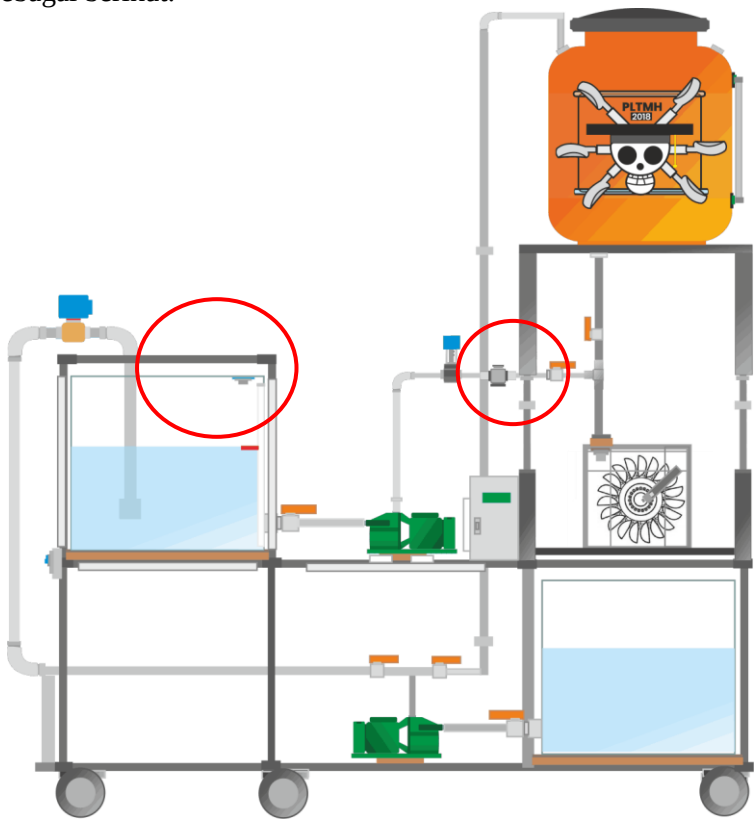
Berikut prosedur pengerjaan alat yang dijelaskan dalam diagram alir sebagai berikut:



Gambar 3.1 Diagram Alir Tugas Akhir

3.2 Desain Pembuatan PLTMH

Pada desain pembuatan PLTMH, dilakukan perancangan untuk membuat sebuah *plant* dengan ukuran skala laboratorium sebagai berikut.



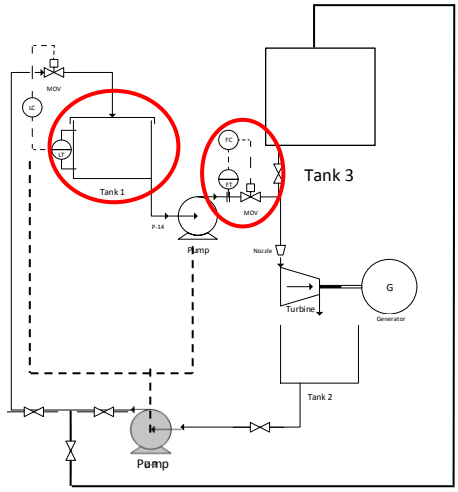
Gambar 3.2 Desain PLTMH

Dari gambar tersebut, sistem monitoring dilakukan pada bagian tangki atas untuk mengetahui ketinggian air dan pipa setelah pompa atas untuk mengetahui aliran air yang mengalir sebelum jatuh mengenai turbin (ditandai dengan lingkaran berwarna merah).

Dan dari gambar diatas, komponen - komponen yang dibutuhkan untuk merakit alat tersebut adalah sebagai berikut.

1. 2 Buah Pompa Air Shimizu
2. 2 Buah Tangki Air ukuran 50x50x50 cm
3. Tandon Air 250L
4. Turbin Pleton
5. Generator
6. Pipa diameter 1”
7. *Motor Valve*
8. *Panel Box*

Semua komponen tersebut dipasang hingga membentuk PLTMH sesuai dengan rancangan pada gambar 3.2. Berikut P&ID dari gambar desain PLTMH.



Keterangan:

Bagian *monitoring*

Gambar 3.3 P&ID PLTMH

3.3 Pemilihan Komponen Untuk Sistem Monitoring

Sebelum dilakukan perancangan sistem monitoring, terlebih dahulu dipilih komponen yang akan digunakan dalam sistem monitoring *flow* dan *level* pada PLTMH sebagai berikut.

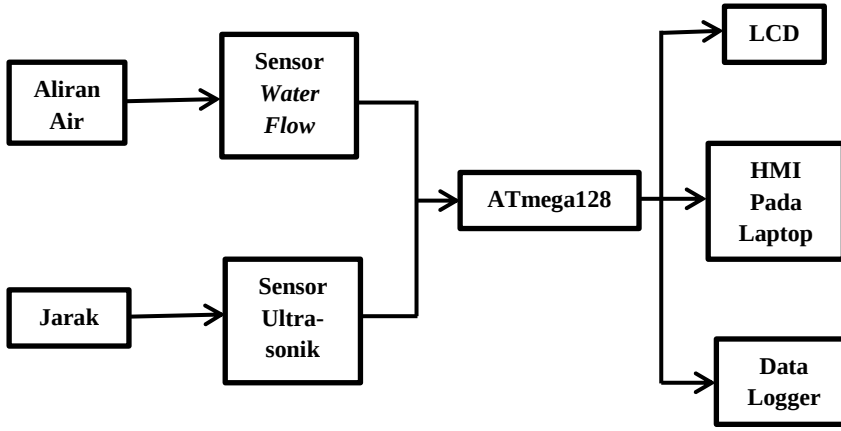
1. Atmega128
2. Sensor Ultrasonik SRF05
3. Sensor *Water Flow* FS400a g1 1”
4. Kabel 5m
5. Kabel Jumper
6. LCD 20x4
7. Modul *Open Log*
8. *Micro* SD *Card* 4gb
9. RTC DS1307
10. Kabel USB tipe B
11. Kabel *Downloader*
12. *Sofware* CodeVision 2.05.03
13. *Software* Visual Studio 2015
14. Khazama AVR Programmer
15. PC (*Portable Computer*)

3.4 Perancangan Sistem *Monitoring*

Pada perancangan sistem *monitoring*, dilakukan perencanaan berupa tahapan – tahapan dalam merancang sistem monitoring sesuai dengan kebutuhan. Berikut *Block Flow Diagram* (BFD) dari sistem monitoring.

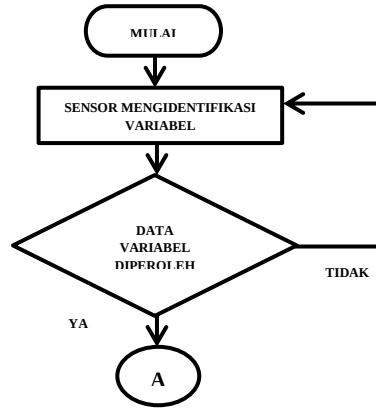
Gambar 3.4 *Block Flow Diagram* Dari *Monitoring*

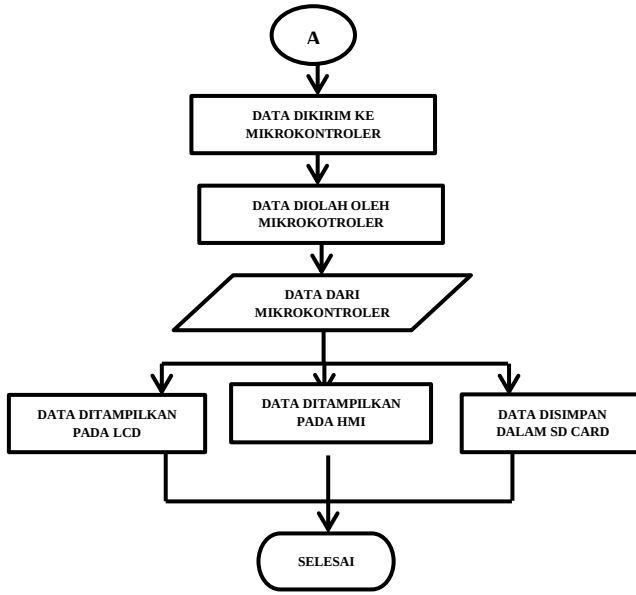
Sensor akan mengidentifikasi variabel sesuai dengan spesifikasi. Data dari sensor tersebut masih berupa analog sehingga akan dikondisikan dan diproses agar saat ditampilkan ke *display* dapat berubah menjadi digital. Dari BFD tersebut, maka dapat dibuat diagram blok dari sistem monitorng *flow* dan *level* pada PLTMH sebagai berikut.



Gambar 3.5 Blok Diagram Sistem *Monitoring Flow* dan *Level* Pada PLTMH

Dari gambar tersebut dapat dilihat bahwa proses akhir dari sistem monitoring adalah *display* pada LCD dan HMI serta penyimpanan data (*data logger*). Pengkondisian dan pemrosesan sinyal untuk kedua variabel berada di ATmega128. Disitulah sinyal analog yang berasal dari kedua sensor akan diubah dalam bentuk digital. Untuk lebih mudah dimengerti, berikut penjelasan proses berjalannya sistem *monitoring flow* dan *level*.



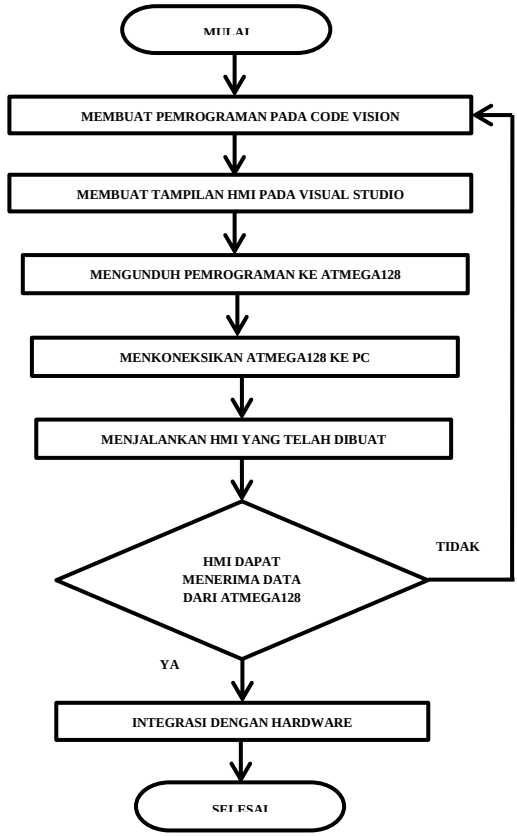


Gambar 3.6 Diagram Alir Penjelasan Sistem *Monitoring* PLTMH

Dari diagram alir tersebut, dijelaskan proses pembacaan variabel oleh sensor hingga diproses didalam mikrokontroler. Untuk variabel *flow*, sensor *water flow* akan menghasilkan data dalam bentuk tegangan. Sedangkan untuk variabel *level*, sensor ultrasonik akan menghasilkan data dalam bentuk arus. Dimana kedua data tersebut akan dikirim ke mikrokontroler untuk diproses agar data dapat berubah menjadi digital. Dari mikrokontroler tersebut, data ditampilkan pada LCD atau HMI (jika disambungkan ke PC) sekaligus dapat bersamaan dilakukan penyimpanan data pada logger untuk mendapatkan hasil pembacaan sensor.

3.4.1 Perancangan HMI (*Human Machine Interface*)

Pada perancangan HMI, digunakan Visual Studio untuk membantu proses pengerjaan HMI. Proses pengerjaan tersebut dapat dijelaskan dalam diagram alir berikut.



Gambar 3.7 Diagram Alir Perancangan HMI

Sebelum memulai pembuatan HMI pada visual studio, pastikan pembuatan pemrograman yang dilakukan pada CodeVision sudah dikerjakan. Pembuatan pemrograman meliputi pembacaan sensor, penampilan pembacaan di LCD, penampilan pembacaan pada komunikasi serial untuk HMI (*Human Machine Interface*), dan penyimpanan *data logger* pada SD Card (isi pemrograman dapat dilihat pada lampiran).

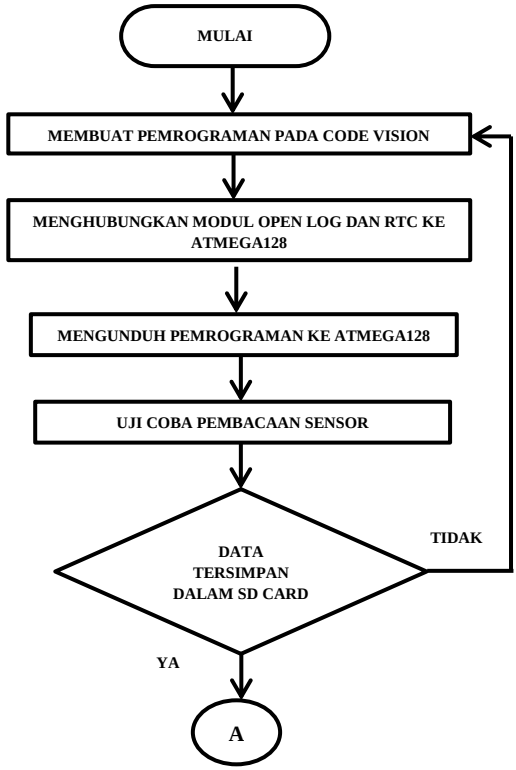
Untuk tampilan HMI (*Human Machine Interface*) dibuat di Visual Studio dengan format .vb (visual basic). Tampilan yang dibuat meliputi seluruh gambar sistem, dan tampilan pembacaan

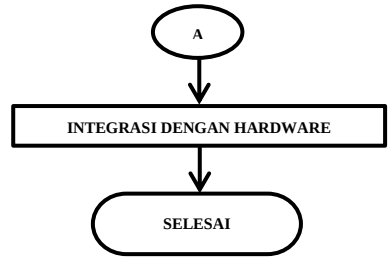
pada bagian sensor untuk *level* dan sensor untuk *flow*. Isi pemrograman HMI dapat dilihat pada lampiran.

Ketika menjalankan program HMI, pemrograman yang telah dibuat pada CodeVision akan diunggahke ATmega128. Dari Atmega tersebut, pemrograman akan diunduh ke PC dengan menggunakan kabel USB tipe B. Kemudian barulah program HMI dapat dijalankan.

3.4.2 Perancangan *Data Logger*

Data logger diperlukan untuk proses penyimpanan data selama proses *monitoring* berjalan. Perancangan *data logger* menggunakan modul *open log* dengan *micro* SD *card* sebesar 4gb. Dan untuk memberi waktu dalam *data logger*, digunakan RTC. DS1307. Berikut proses perancangan *data logger* dalam PLTMH.





Gambar 3.8 Diagram Alir Perancangan *Data Logger*

3.5 Perakitan Kabel Untuk Sistem *Monitoring* Pada *Plant*

Pada perakitan kabel, dilakukan penyambungan semua komponen ke mikrokontroler. Berikut sambungan pin sensor, LCD, HMI, dan *data logger* pada port di atmega:

a. Sensor Ultrasonik SRF05

Tabel 3.1 Konfigurasi Pin Sensor Ultrasonik

Pin Pada Sensor	Port Pada Mikrokontroller
Trig	PORTA.1
Echo	PORTA.3
Gnd	PORTA.Gnd
Vcc	PORTA.5v

b. Sensor *Water Flow* FS400a 1”

Tabel 3.2 Konfigurasi Pin Sensor *Water Flow*

Pin Pada Sensor	Port Pada Mikrokontroller
Kabel hitam	PORTD.Gnd
Kabel kuning	POTRD.6
Kabel merah	PORTD.Vcc

c. Modul *Open Log* SD Card

Tabel 3.3 Konfigurasi Pin *Data Logger*

Pin Pada Modul	Port Pada Mikrokontroller
Gnd	Gnd
Vcc	+5v
Tx	POTRD.2
Rx	PORTD.3

d. RTC DS1307

Tabel 3.4 Konfigurasi Pin RTC

Pin Pada RTC	Port Pada Mikrokontroller
Gnd	Gnd
Vcc	+5v
Sda	PORTD.1
Scl	PORTD.0

e. Kabel USB Untuk HMI

Tabel 3.5 Konfigurasi Pin USB

USB	Mikrokontroller
Port kecil	PORT USB

Mikrokontroler diletakkan pada *panel box* dan semua sambungan tersebut akan ditata pada panel box juga. Untuk sambungan pada sensor yang jaraknya cukup jauh, dapat dibantu dengan sambungan kabel yang telah disolder dengan kabel *jumper*. Penataan kabel yang jauh juga dapat dibantu oleh kabel *duct* agar terlihat lebih rapi dan meminimalisir terkena tetesan air.

3.6 Integrasi Sistem *Monitoring* Dengan *Hardware*

Integrasi diperlukan agar sistem monitoring yang telah dibuat dan *hardware*, yaitu perakitan kabel dapat berjalan dengan baik. Ketika tidak terjadi error pada saat penyambungan *hardware* dengan *software*, maka proses integrasi berhasil dan dapat dilakukan pengujian. Error yag terjadi biasanya data tidak dapat muncul di LCD, tidak dapat terkoneksi dengan HMI, atau data tidak terbaca dalam *logger*.

3.7 Pengujian Sistem *Monitoring*

Pada pengujian sistem monitoring dilakukan pengujian untuk mendapatkan respon dari mikrokontroler yang terdapat pada rancang bangun PLTMH. Dimana mikrokontroler tersebut telah terintegrasi dengan sensor yang ada pada rancang bangun PLTMH. Sehingga hasil pembacaan yang didapat dari sensor akan

diolah oleh mikrokontroler untuk menghasilkan sebuah sistem monitoring.

Jika hasil uji tidak berhasil atau tidak dapat terbaca, maka akan dicek pada proses perancangan sistem monitoring. Apakah ada kesalahan sistem sehingga mengakibatkan hasil uji sistem monitoring tidak berhasil. Parameter keberhasilan pengujian ini adalah data yang muncul pada LCD dan HMI menampilkan haisl yang sama. Data yang terbaca juga *real time* dan sesuai dengan variabel yang diinginkan. Serta data dapat tersimpan dalam *data logger*.

“Halaman ini sengaja dikosongkan”

.

BAB IV
HASIL DAN PEMBAHASAN

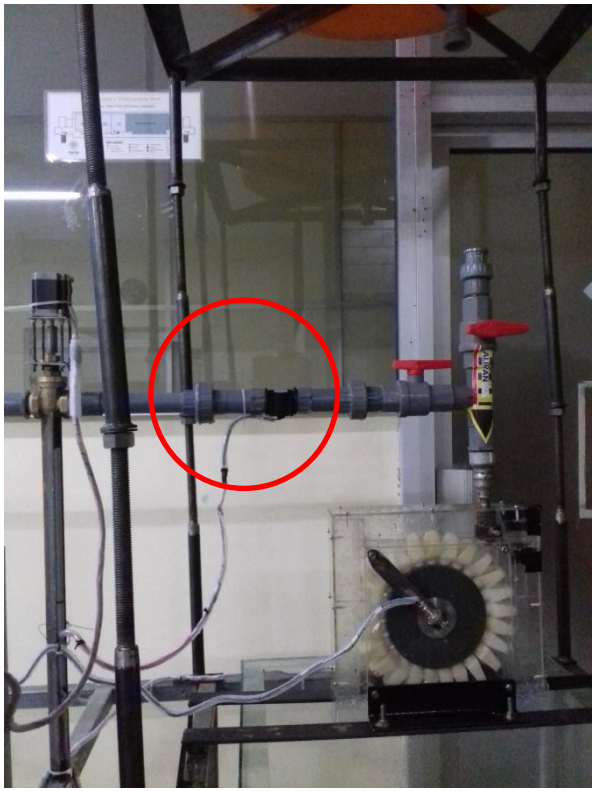
4.1 Hasil Perancangan Sistem

Berikut ini merupakan hasil perancangan sistem monitoring *flow* dan *level* pada Pembangkit Listrik Tenaga Mikro Hidro (PLTMH).

Gambar 4.1 Hasil Desain PLTMH

Pada gambar 4.1 merupakan gambar keseluruhan Pembangkit Listrik Tenaga Mikro Hidro (PLTMH). Pada PLTMH tersebut sistem *monitoring level* dilakukan pada tangki

atas dan sistem *monitoring flow* pada aliran setelah pompa atas. Sensor ultrasonik diletakkan menempel pada ujung tangki paling atas dan sensor *water flow* diletakkan setelah MOV.

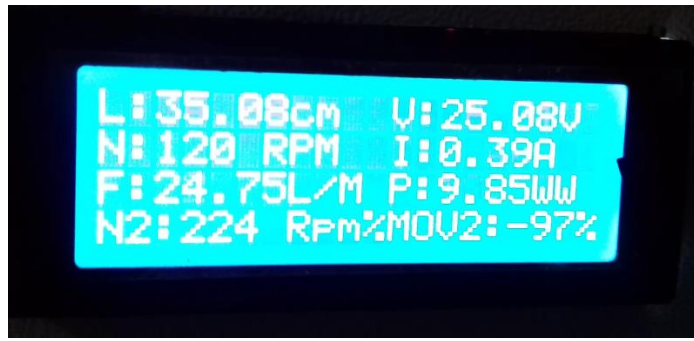


Gambar 4.2 Peletakkan Sensor *Flow*

Peletakkan sensor *water flow* ditandai dengan lingkaran berwarna merah. Dimana sensor tersebut dihubungkan dengan pipa PVC sebesar 1”. Dan berikut gambar peletakkan sensor ultrasonik yang ditandai dengan lingkaran berwarna merah juga.

Gambar 4.3 Peletakkan Sensor *Level*

Hasil pembacaan sistem monitoring pada *plant* dapat langsung dilihat pada LCD yang menempel pada *panel box* seperti gambar 4.4.



Gambar 4.4 Tampilan Pembacaan Sensor Pada LCD

Pada LCD tersebut menampilkan 8 pembacaan data. Dimana *flow* diwakilkan dengan inisial huruf F dengan satuan L/M (liter per menit), dan *level* diwakilkan dengan inisial huruf L dengan satuan cm. Dan 6 pembacaan yang lain yaitu arus, tegangan, daya, besarnya RPM yang dihasilkan oleh turbin, RPM yang dihasilkan oleh generator, serta besarnya bukaan MOV yang memepengaruhi besarnya debit aliran air pada PLTMH.

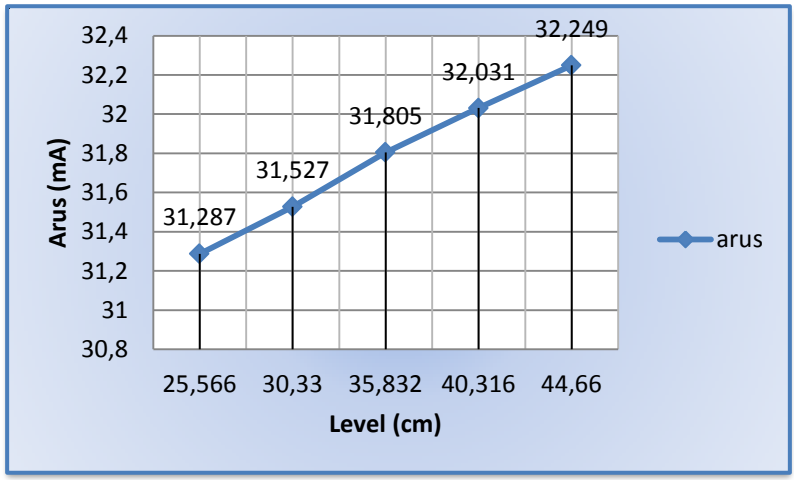
4.1.1 Hasil Uji Pembacaan *Level*

Pada pengujian pembacaan level, akan dilakukan variasi tinggi air pada tangki yang kemudian akan dibaca oleh sensor. Hasil pembacaan tersebut akan diketahui besarnya arus yang terjadi pada saat pembacaan berlangsung.

Tabel 4.1 Hasil Pembacaan Sensor Ultrasonik

No.	Variasi Level (cm)	Pembacaan Sensor *Level* (cm)	Arus (mA)
1.	45	44,66	32,24987
2.	40	40,316	32,031
3.	35	35,832	31,805
4.	30	30,33	31,52796
5.	25	25,566	31,28796

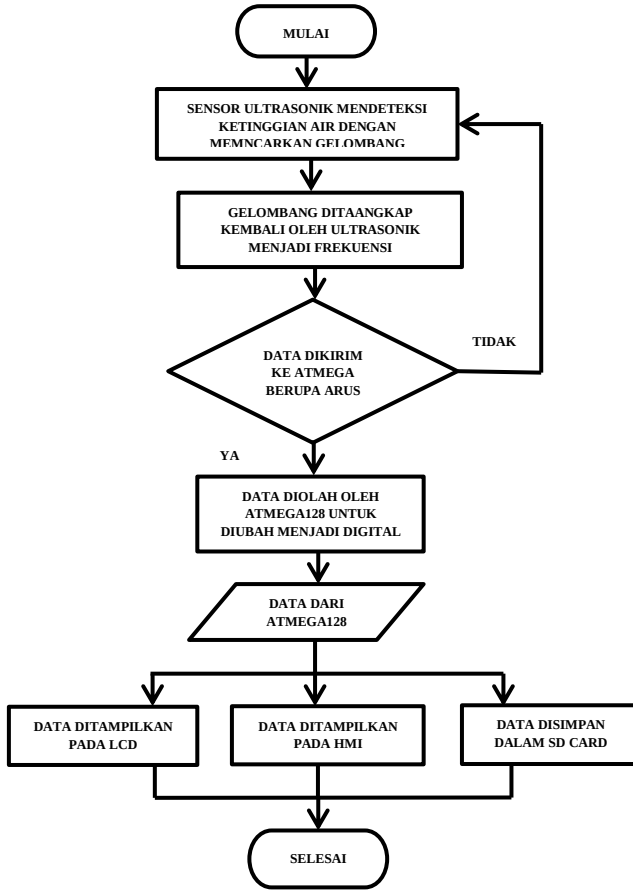
Dari tabel 4.1 tersebut dilakukan 5 kali pengambilan data dengan 5 variasi yang berbeda. Dari setiap variasi dapat dilihat hasil pembacaannya pada kolom pembacaan sensor *level* (cm). Dari pembacaan tersebut dapat diketahui arus yang mengalir ketika terjadi pembacaan *level* oleh sensor ultrasonik. Berikut grafik pembacaan sensor *level* (cm) terhadap arus (mA).



Gambar 4.5 Grafik Perbandingan Pembacaan *Level*

Pada gambar 4.5 merupakan grafik hasil uji pembacaan ketinggian air (*level*) oleh sensor ultrasonik. Pada grafik tersebut dapat dilihat besarnya *level* terhadap arus. Arus yang dihasilkan berbanding lurus dengan besarnya *level* yang terbaca oleh sensor. Yaitu ketika *level* semakin tinggi maka arus yang dihasilkan akan meningkat pula. Arus tersebut merupakan arus yang berasal dari pendeteksian sensor ultrasonik terhadap jarak benda didepannya. Arus tersebut kemudian dikirim ke ATmega untuk dikondisikan dan diproses agar dapat diubah menjadi digital. Yaitu dalam satuan cm.

Untuk lebih jelasnya, berikut diagram alir penjelasan proses pembacaan sensor ultrasonik sehingga dapat menghasilkan data pembacaan seperti pada tabel 4.1.



Gambar 4.6 Proses Pembacaan *Level* Oleh Sensor Ultrasonik

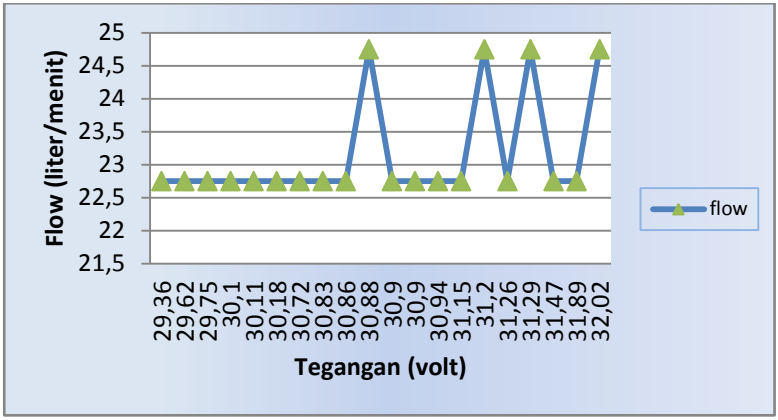
4.1.2 Hasil Uji Pembacaan *Flow*

Pada pembacaan sensor *flow* dilakukan pengambilan data sebanyak 20 kali dengan *set point* tegangan sebesar 30 volt. Dari pembacaan tersebut dapat dilihat besarnya debit yang mengalir dan besarnya tegangan yang dihasilkan dari *set point* tersebut yang berasal dari putaran turbin. Berikut hasil data pembacaan sensor *water flow*.

Tabel 4.2 Hasil Pembacaan Sensor *Water Flow*

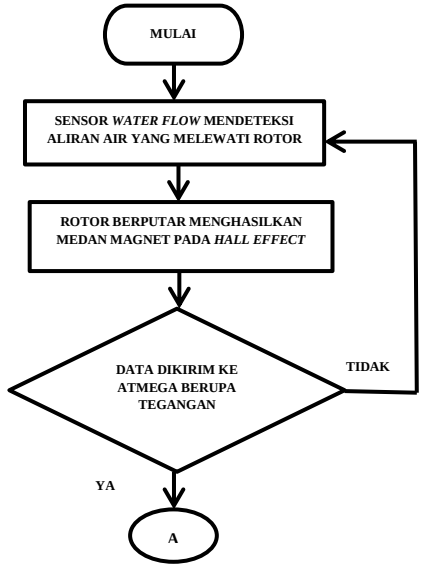
Set Point (volt)	Pembacaan Sensor (liter/menit)	Menghasilkan Tegangan (volt)
30	22,75	30,72
30	22,75	30,18
30	22,75	29,62
30	22,75	29,36
30	24,75	31,20
30	22,75	31,26
30	22,75	31,89
30	24,75	31,29
30	22,75	30,90
30	22,75	30,11
30	22,75	31,15
30	22,75	30,88
30	22,75	32,02
30	22,75	31,47
30	22,75	29,75
30	24,75	30,83
30	22,75	30,10
30	22,75	30,86
30	22,75	30,94
30	24,75	30,90

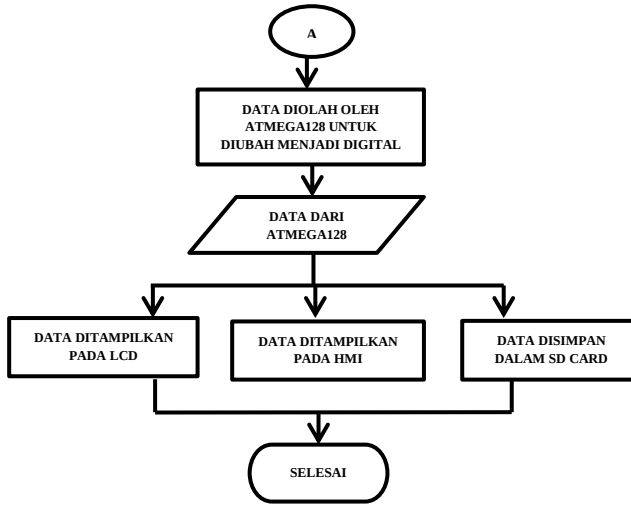
Dari tabel tersebut dapat dilihat besarnya debit yang mengalir dengan *set point* tegangan yang dihasilkan sebesar 30 volt. Dan juga dapat dilihat besarnya tegangan yang terbaca dari *set point* tersebut. Dapat dilihat bahwa pembacaan sensor *water flow* pada saat tersebut cenderung stabil yaitu berada di nilai debit sebesar 22,75 liter/menit. Meskipun besarnya debit cenderung stabil, namun tegangan yang dihasilkan oleh turbin masih fluktuatif di angka yang berdekatan dengan 30. Berikut grafik yang dihasilkan dari tabel 4.2.



Gambar 4.7 Grafik Pembacaan *Flow* Terhadap Tegangan

Dari gambar tersebut dapat dilihat bahwa garis paling bawah menunjukkan nilai 22,75 liter/menit dan titik yang paling atas menunjukkan angka 24,75 liter/menit. Berikut penjelasan proses pembacaan sensor *water flow* sehingga dapat menghasilkan data pembacaan seperti pada tabel 4.2.



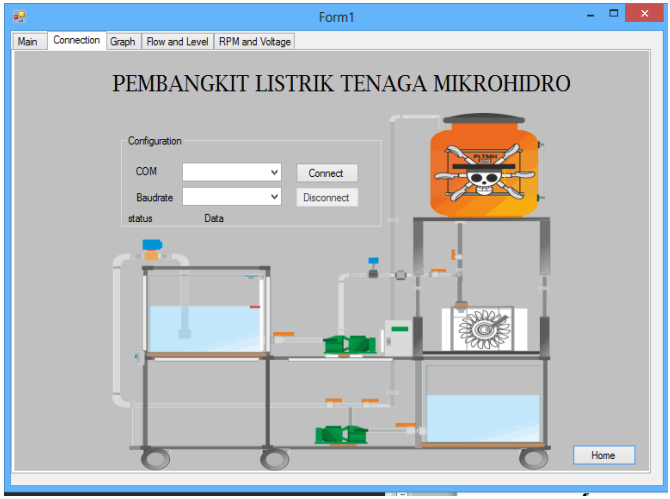


Gambar 4.8 Proses Pembacaan *Flow* Oleh Sensor *Waterflow*

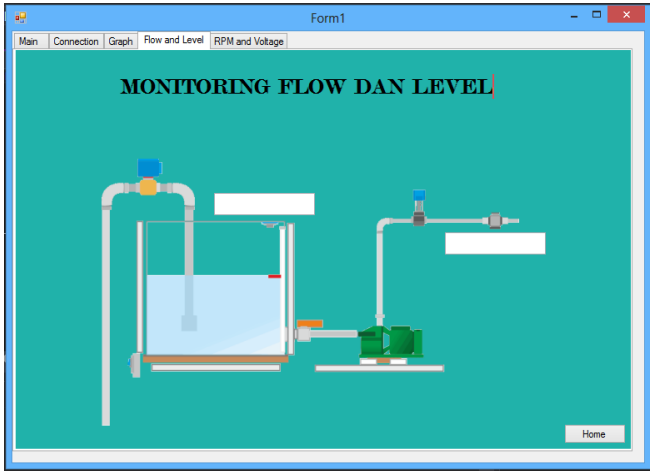
4.1.3 Hasil Perancangan HMI (*Human Machine Interface*)

Pada sistem monitoring *flow* dan *level* pada PLTMH, data yang terbaca dapat ditampilkan pada HMI. Berikut hasil perancangan *Human Machine Interface* (HMI) yang telah dibuat pada Visual Studio.

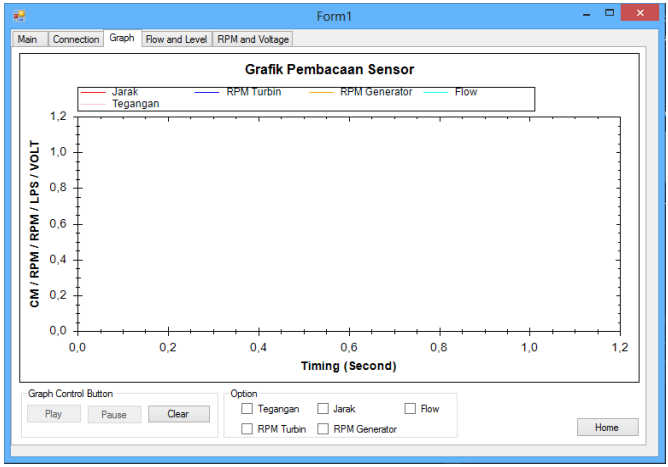
Gambar 4.9 Tampilan Utama HMI



Gambar 4.10 Tampilan Desain PLTMH



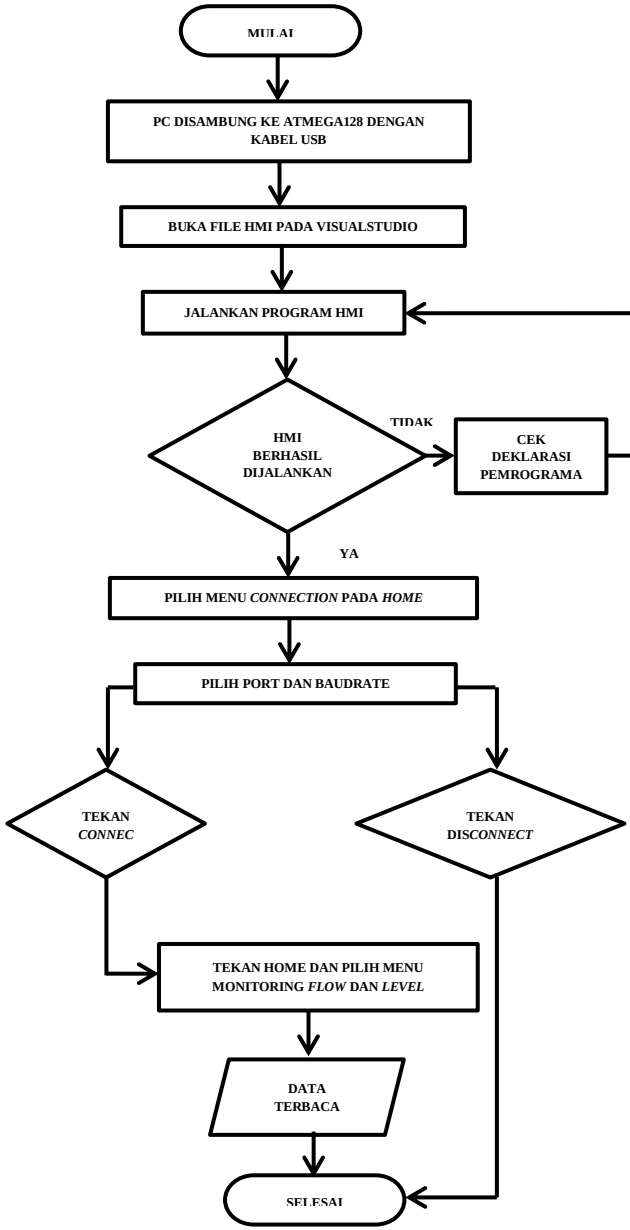
Gambar 4.11 Tampilan Monitoring *Flow* dan *Level*



Gambar 4.12 Tampilan Grafik Pada HMI

Pada gambar 4.9 merupakan tampilan utama HMI ketika dijalankan. Tampilan tersebut memunculkan menu pilihan sistem yang ingin dijalankan. Namun sebelumnya pastikan PC telah terkoneksi dengan mikrokontroler menggunakan USB tipe B dan mikrokontroler telah memiliki pemrograman yang siap diunduh untuk proses berjalannya pembacaan data oleh sensor pada HMI. Barulah memilih port dan *baudrate* yang tersedia pada menu *connection* dan hasilnya akan seperti pada gambar 4.10. Setelah terkoneksi, dapat ditekan tombol home dan memilih menu *monitoring flow* dan *level*. Kemudian tampilan akan berubah seperti pada gambar 4.11. Untuk melihat grafik dapat kembali ke menu *home* dan memilih menu grafik. Maka tampilan akan berubah seperti gambar 4.12. Lalu centang variabel yang ingin ditampilkan pada grafik. Setelah itu tekan tombol *play*.

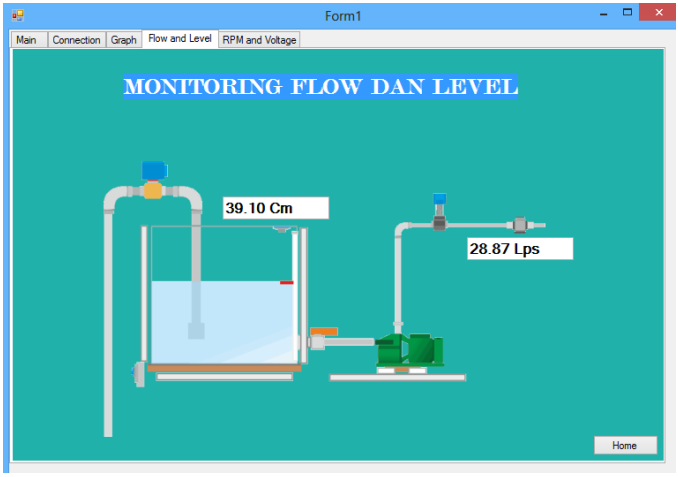
Untuk lebih jelasnya, berikut proses berjalannya HMI agar dapat menampilkan pembacaan sensor pada PLTMH.



Gambar 4.13 Proses Menjalankan HMI

Setelah melalui semua proses tersebut maka tampilan HMI pada pembacaan monitoring flow dan level akan nampak seperti pada gambar 4.14 dan 4.15.

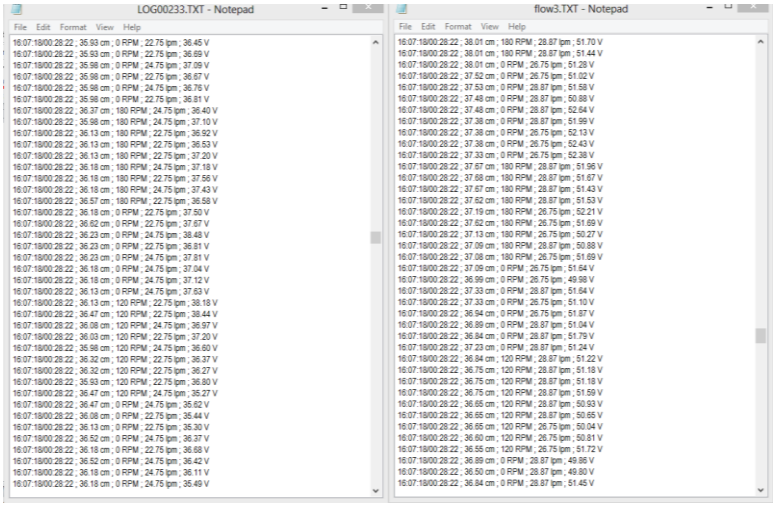
Gambar 4.14 Tampilan HMI Yang Telah Terkoneksi



Gambar 4.15 Tampilan HMI Pada Saat Uji *Flow* dan *Level*

4.1.4 Hasil Perancangan *Data Logger*

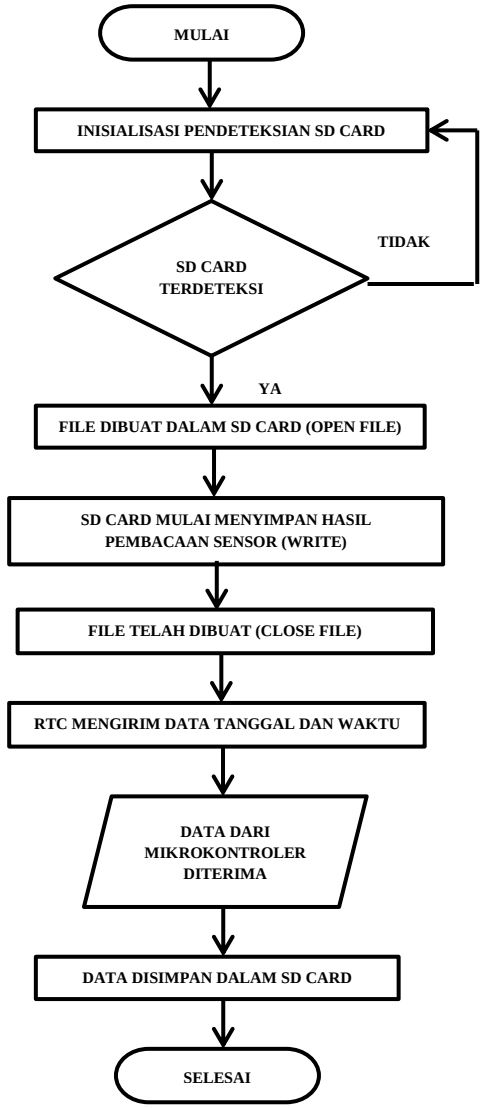
Data yang telah diproses dalam mikrokontroler juga otomatis akan disimpan oleh modul *open log* dalam SD *Card*. Berikut hasil penyimpanan data logger yang telah tersimpan dalam SD card. File tersebut tersimpan dalam format .txt.



Gambar 4.16 Hasil Penyimpanan Data Pada SD *Card*

Penyimpanan data tersebut telah menyimpan 4 data yang berbeda. Dimana data tersebut dipisahkan oleh tanda ";" dan dibedakan dengan diberi satuan dibelakang angka. Untuk pembacaan *level* ditandai dengan satuan cm. Dan untuk *flow* ditandai dengan satuan lpm (liter per menit).

Data disebelah kiri merupakan hasil data dengan mengatur set point tegangan sebesar 38 volt. Dan sebelah kanan merupakan hasil pembacaan data dengan kondisi MOV terbuka penuh dengan hasil rata – rata tegangan sebesar lebih dari 50 volt. Berikut penjelasan proses penyimpanan *data logger*.



Gambar 4.17 Proses Inisiasi *Data Logger*

Dari penjelasan tersebut, modul open log yang telah terdeteksi akan menyala. Dan pada saat proses penyimpanan data

berlangsung, modul open log akan menyala kedip – kedip yang menandakan bahwa proses penyimpanan data sedang berjalan.

4.1.5 Hasil Validasi Pembacaan Sensor

Untuk mengetahui bahwa hasil pembacaan sensor sesuai dengan pembacaan yang sesungguhnya, maka dilakukan validasi pembacaan sensor. Validasi sensor dilakukan untuk mengetahui tingkat akurasi dan *error* dari pembacaan.

Tabel 4.3 Hasil Validasi Pembacaan Sensor *Water Flow*

No	*Error*	Akurasi
1.	1,645 %	98,355 %

Hasil validasi pembacaan sensor *water flow* tersebut didapat dengan data dan perhitungan yang tertera pada Lampiran F (Karakteristik Statik Pembacaan Sensor *Water Flow*).

Tabel 4.4 Hasil Validasi Pembacaan Sensor Ultrasonik

No	*Error*	Akurasi
1.	0,974 %	99,026 %

Hasil validasi pembacaan sensor ultrasonik tersebut didapat dengan data dan perhitungan yang tertera pada Lampiran G (Karakteristik Statik Pembacaan Sensor Ultrasonik).

4.2 Pembahasan

Pengambilan data untuk pengujian *flow* dan *level* dilakukan pada saat yang bersamaan. Ketika *plant* menyala maka proses mekanik dan elektrik juga berjalan. Pompa akan mendorong air melewati pipa sampai jatuh mengenai turbin. Pada saat itulah dapat dilihat besarnya *level* pada tangki atas dan besarnya aliran yang melalui *water flow*. Besarnya *level* dan *flow* yang terbaca dapat dilihat melalui LCD pada *panel box*, tampilan HMI pada

PC, atau dapat dilihat melalui data logger yang telah tersimpan didalam *SD card*.

Sebelum menyalakan PLTMH, pastikan semua pin dari setiap komponen terpasang dengan sesuai pada ATmega128. Karena kesalahan pemasangan dapat mempengaruhi pembacaan sensor. Lalu pastikan juga bahwa pemrograman yang telah dibuat untuk diunggah kedalam ATmega128 tidak memiliki error. Setelah itu untuk menjalankan HMI pastikan kabel USB tipe B menancap pada PC. Pemilihan USB tipe B sebagai penghubung komunikasi serial antara PC dan mikrokontroler karena bentuknya yang lebih praktis dan memiliki banyak variasi panjang kabel sesuai kebutuhan. Untuk kecepatan pengiriman data antara beberapa kabel sebenarnya tidak ada bedanya. Namun perbedaan antara USB tipe B, kabel RTS232, atau usb TTL lebih terletak pada kepraktisan penggunaannya. Karena untuk menggunakan usb TTL memelurkan sambungan beberapa port pada atmega dan perlu sambungan kabel yang panjang agar dapat menjangkau PC yang jauh dari *panel box*.

Untuk pengujian data *level* diambil sebanyak 5 kali dengan 5 variasi ketinggian yang berbeda. Dari hasil pembacaan tersebut akan dicari arus yang mengalir ketika sensor ultrasonik mengirim sinyal ke ATmega128. Dari hasil pengujian tersebut dapat dilihat bahwa besarnya arus dan besarnya ketinggian berbanding lurus. Yaitu ketika level turun maka arus juga ikut turun. Lalu dalam pengujian *flow* diambil data sebanyak 20 kali dengan *set point* tegangan yang dihasilkan sebesar 30 volt. Hasil data pembacaan menunjukkan bahwa pembacaan sensor *water flow* cukup stabil karena minimnya perubahan pembacaan yang terjadi. Besarnya debit aliran ini juga dibandingkan dengan besarnya tegangan yang dihasilkan oleh putaran turbin. Dan hasilnya cenderung stabil.

Sensor yang telah mendeteksi objek akan mengirim sinyal berupa analog ke mikrokontroler. Dari mikrokontroler tersebut,

sinyal akan dikondisikan dan diproses untuk berubah menjadi digital. Sinyal digital itulah yang akan dikirim ke LCD untuk dapat ditampilkan menjadi bentuk angka. Selain itu, sinyal digital yang telah diproses dalam mikrokontroler juga dapat ditampilkan melalui HMI dengan adanya komunikasi serial menggunakan USART. Mikrokontroler yang telah terhubung ke PC akan mengirimkan data melalui kabel USB yang sebelumnya sudah dikoneksikan. Secara bersamaan pula, data dari mikrokontroler juga akan dikirim ke modul *open log* untuk disimpan dalam SD *card* sebagai penyimpanan data dalam *data logger*.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian dan perancangan sistem monitoring *flow* dan *level* pada Pembangkit Listrik Tenaga Mikro Hidro (PLTMH), maka didapatkan kesimpulan sebagai berikut.

1. Telah berhasil dirancang sistem *monitoring flow* dan *level* pada Pembangkit Listrik Tenaga Mikro Hidro (PLTMH) yang dapat menyimpan dan menampilkan hasil data pembacaan secara *real time* dengan tingkat akurasi sebesar 99,026% untuk pembacaan *level* dan 98,355% untuk pembacaan *flow*.
2. Telah berhasil dilakukan proses penyimpanan data dengan menggunakan modul *open log* pada SD Card dengan format .txt serta hasil data dapat ditampilkan pada LCD 20 x4 dan ditampilkan pada HMI.

5.2 Saran

Saran yang dapat diberikan untuk tugas akhir ini adalah sebagai berikut.

1. Perlu dilakukan pengembangan kedepannya untuk sistem HMI agar dapat melakukan proses pengendalian dan kontrol didalamnya.

“Halaman ini sengaja dikosongkan”

DAFTAR PUSTAKA

(2012). Dipetik July 7, 2018, dari CV. BINTANG JAYA TEKNIK: http://www.bintangjayatenik.com

Apriansyah, F., Rusdinar, A., & Darlis, D. (2016, April). RANCANG BANGUN SISTEM PEMBANGKIT LISTRIK MIKROHIDRO (PLTMH) PADA PIPA SALURAN PEMBUANGAN AIR HUJAN VERTIKAL. *e-Proceeding of Engineering*, 59.

Arifin, S., & Fathoni, A. (2014). Pemanfaatan Pulse Width Modulation Untuk Mengontrol Motor (Studi Kasus Robot Otomatis Dua Deviana. *Jurnal Ilmiah Teknologi dan Informasi ASIA*, Vol.8 No.2.

Arismunandar, W. (1997). *Penggerak Mula Turbin.* Bandung: ITB.

Fitriandi, A., Komalasari, E., & Gusmedi, H. (2016, Mei). Rancang Bangun Alat Monitoring Arus dan Tegangan Berbasis. *Jurnal Rekayasa dan Teknologi Elektro*, Volume 10 No.2.

Fitriandi, A., Komalasari, E., & Gusmedi, H. (2016, Mei). Rancang Bangun Alat Monitoring Arus dan Tegangan Berbasis Mikrokontroler dengan SMS Gateway. *Jurnal Rekayasa dan Teknologi Elektro, 10*, 89.

Fitriandi, A., Komalasari, E., & Gusmedi, H. (2016). Rancang Bangun Alat Monitoring Arus dan Tegangan Berbasis Mikrokontroler dengan SMS Gateway. *Jurnal Rekayasa dan Teknologi Elektro*, 89.

Fitriandi, A., Komalasari, E., & Gusmedi, H. (2016). Rancang Bangun Alat Monitoring Arus dan Tegangan Berbasis Mikrokontroler dengan SMS Gateway. *Jurnal Rekayasa dan Teknologi Elektro*, Volume 10 No.2.

Fox, R., & Mc. Donald, A. (1994). *Introduction to Fluid Mechanics, fourth edition.* Canada: Inc. Canada.

Ismono, H. (1999). *Perencanaan Turbin Air Tipe Cross Flow untuk Pembangkit Listrik Tenaga Mikrohidro di Institut Teknologi Nasional Malang.* Malang: Institut Teknologi Nasional Malang.

Karakteristik Statik Elemen Sistem Pengukuran. (t.thn.). *Modul Ajar Praktikum Sistem Pengukuran & Kalibrasi*, 2 - 11.

Nadiya, S. (2016). Pemanfaatan Sensor Ultrasonik Dalam Pengukuran Debit Air Pada Saluran Irigasi Berbasis Mikrokontroler Atmega8535 Menggunakan Media Penyimpanan SD Card. *Jurnal Jurusan Fisika Universitas Lampung*.

Prayogo, E. (2003). *Teknologi Mikrohidro dalam Pemanfaatan Sumber Daya Air*. Makassar: Semiloka Produk - Produk Penelitian Departement Kimpraswill.

Purnama, A. (2012). *Belajar Elektronika Dasar*. Dipetik Januari 18, 2018, dari Definisi dan Fungsi Sensor Efel Hall: www.elektronika-dasar.wed.id

Sihaloho, D. L. (2017). RANCANG BANGUN ALAT UJI MODEL SISTEM PEMBANGKIT LISTRIK TENAGA MIKRO HIDRO (PLTMH) MENGGUNAKAN TURBIN ALIRAN SILANG. 6.

Sukamta, S., & Kusmantoro, A. (2013, Juli - Desember). Perencanaan Pembangkit Listrik Tenaga Mikro Hidro (PLTMH) Jantur Tabalas Kalimantan Timur. *Jurnal Teknik Elektro*, Vol. 5 No. 2.

Termas Media. (t.thn.). Dipetik April 1, 2018, dari Pengertian Database: http://www.termasmedia.com/lainnya/software/69-pengertian-database.html

Ujang, H. d. (2007). *Desain, manufacturing dan instalasi turbin propeller open flume Ø 125 Mm di CV Cihanjuang Inti Teknik Cimahi Jawa Barat.* Bogor: Fakultas Teknologi Pertanian IPB.

Unikom. (t.thn.). Dipetik January 15, 2018, dari http://elib.unikom.ac.id/files/disk1/452/jbptunikompp-gdl-bennymuhar-22559-2-unikom_b-i.pdf

LAMPIRAN A
(*Datasheet* ATmega128)

Features

- High-performance, Low-power Atmel®AVR®8-bit Microcontroller
- Advanced RISC Architecture
  - 133 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
  - Fully Static Operation
  - Up to 16MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 128Kbytes of In-System Self-programmable Flash program memory
  - 4Kbytes EEPROM
  - 4Kbytes Internal SRAM
  - Write/Erase cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Up to 64Kbytes Optional External Memory Space
  - Programming Lock for Software Security
  - SPI Interface for In-System Programming
- QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix acquisition
  - Up to 64 sense channels
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Output Compare Modulator
  - 8-channel, 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels
    - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USARTs
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
  - Software Selectable Clock Frequency
  - ATmega103 Compatibility Mode Selected by a Fuse
  - Global Pull-up Disable
- I/O and Packages
  - 53 Programmable I/O Lines
  - 64-lead TQFP and 64-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V ATmega128L
  - 4.5 - 5.5V ATmega128
- Speed Grades
  - 0 - 8MHz ATmega128L
  - 0 - 16MHz ATmega128

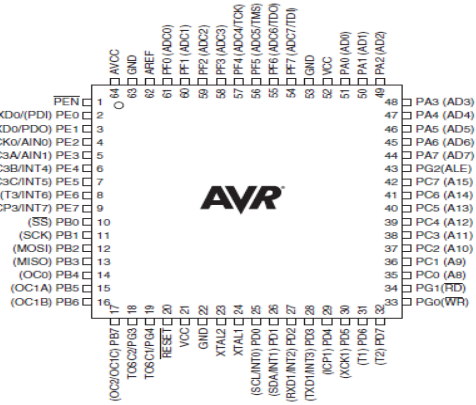
8-bit Atmel Microcontroller with 128KBytes In-System Programmable Flash

ATmega128
ATmega128L

Rev. 2467X–AVR–06/11

Pin Configurations

Figure 1. Pinout ATmega128



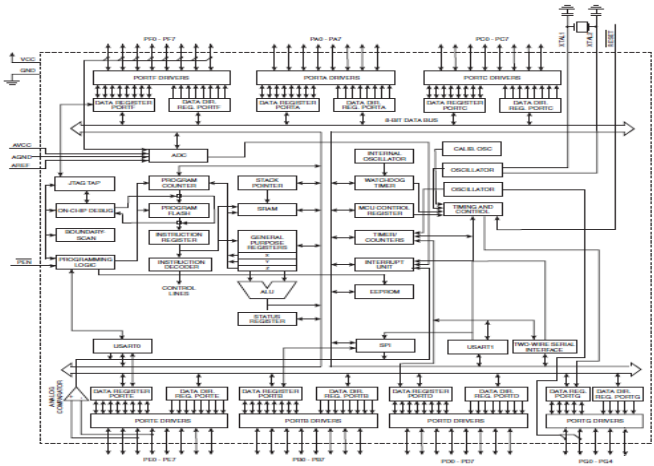
Note: The Pinout figure applies to both TQFP and MLF packages. The bottom pad under the QFN/MLF package should be soldered to ground.

Overview

The Atmel® AVR® ATmega128 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega128 achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The Atmel® AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega128 provides the following features: 128Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 4Kbytes EEPROM, 4Kbytes SRAM, 53 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), four flexible Timer/Counters with compare modes and PWM, 2 USARTs, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with Internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the Crystal/Resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

Atmel offers the QTouch® library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS™) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop and debug your own touch applications.

The device is manufactured using Atmel's high-density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega128 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega128 device is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

ATmega103 and ATmega128 Compatibility

The ATmega128 is a highly complex microcontroller where the number of I/O locations supersedes the 64 I/O locations reserved in the AVR instruction set. To ensure backward compatibility with the ATmega103, all I/O locations present in ATmega103 have the same location in ATmega128. Most additional I/O locations are added in an Extended I/O space starting from $60 to $FF, (i.e., in the ATmega103 internal RAM space). These locations can be reached by using LD/LDS/LDD and ST/STS/STD instructions only, not by using IN and OUT instructions. The relocation of the internal RAM space may still be a problem for ATmega103 users. Also, the increased number of interrupt vectors might be a problem if the code uses absolute addresses. To solve these problems, an ATmega103 compatibility mode can be selected by programming the fuse M103C. In this mode, none of the functions in the Extended I/O space are in use, so the internal RAM is located as in ATmega103. Also, the Extended Interrupt vectors are removed.

LAMPIRAN B
(*Datasheet* Sensor *Water Flow* FS400a)

G1" Water Flow Sensor

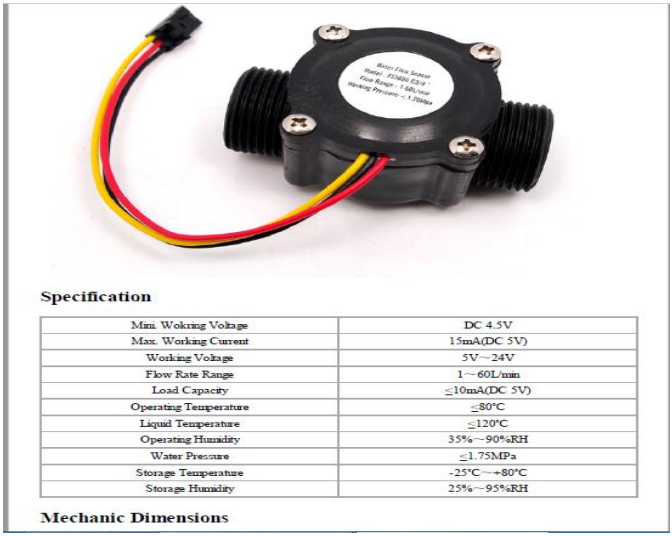
From Wiki

Contents

Introduction

Water flow sensor consists of a plastic valve body, a water rotor, and a hall-effect sensor. When water flows through the rotor, rotor rolls. Its speed changes with different rate of flow. The hall-effect sensor outputs the corresponding pulse Signal.

Model:SEN02141B (http://www.seeedstudio.com/depot/g34-water-flow-sensor-p-1083.html?cPath=144_151)



Specification

Mini. Wokring Voltage	DC 4.5V
Max. Working Current	15mA(DC 5V)
Working Voltage	5V～24V
Flow Rate Range	1～60L/min
Load Capacity	≤10mA(DC 5V)
Operating Temperature	≤80°C
Liquid Temperature	≤120°C
Operating Humidity	35%～90%RH
Water Pressure	≤1.75MPa
Storage Temperature	-25°C～+80°C
Storage Humidity	25%～95%RH

Mechanic Dimensions

Sensor Components

No.	Name	Quantity	Material	Note
1	Valve body	1	PA66+33%glass fiber	
2	Stainless steel bead	1	Stainless steel SUS304	
3	Axis	1	Stainless steel SUS304	
4	Impeller	1	POM	
5	Ring magnet	1	Ferrite	
6	Middle ring	1	PA66+33%glass fiber	
7	O-seal ring	1	Rubber	
8	Electronic seal ring	1	Rubber	
9	Cover	1	PA66+33%glass fiber	
10	Screw	4	Stainless steel SUS304	
11	Cable	1	1007 24AWG	

Usage Example

Note: This example is abstracted from the forum, which was done by Charles Gantt. Thanks for his contribution.Let's see how it works.

Reading Water Flow rate with Water Flow Sensor

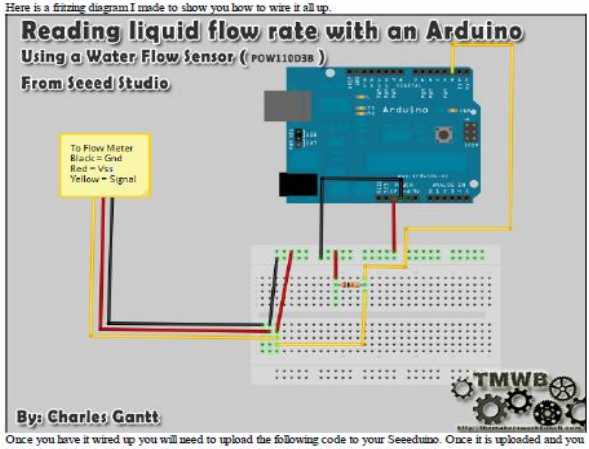
This is part of a project I have been working on and I thought I would share it here since there have been a few threads on how to read water flow rate in liters per hour using the Water Flow Sensor found in the Seeed Studio Depo. It uses a simple rotating wheel that pulses a hall effect sensor. By reading these pulses and implementing a little math, we can read the liquids flow rate accurate to within 3%. The threads are simple G3/4 so finding barbed ends will not be that hard.

Hardware Installation

You will need Seeeduino / Arduino ,Water Flow Sensor,10K resistor,a breadboard and some jumper wires.

Wiring up the Water Flow Sensor is pretty simple. There are 3 wires: Black, Red, and Yellow. Black to the Seeeduino's ground pin Red to Seeeduino's 5v pin The yellow wire will need to be connected to a 10k pull up resistor.and then to pin 2 on the Seeeduino.

Here is a fritzing diagram I made to show you how to wire it all up.



Once you have it wired up you will need to upload the following code to your Seeeduino. Once it is uploaded and you have some fluid flowing through the Water Flow Sensor, you can open the serial monitor and it will display the flow rate, refreshing every second.

Programming

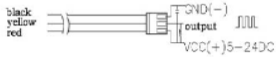
```
// reading liquid flow rate using Seeeduino and Water Flow Sensor from Seeedstudio.com
// Code adapted by Charles Gantt from PC Fan RPM code written by Crenn @thebestcasescenario
// http://themakersworkbench.com http://thebestcasescenario.com http://seeedstudio.com

volatile int NbTopsFan; //measuring the rising edges of the signal
int Calc;
int hallsensor = 2;    //The pin location of the sensor

void rpm ()     //This is the function that the interupt calls
{
  NbTopsFan++;  //This function measures the rising and falling edge of the

hall effect sensors signal
}
// The setup() method runs once, when the sketch starts
```

```
void setup() //
{
  pinMode(hallsensor, INPUT); //initializes digital pin 2 as an input
  Serial.begin(9600); //This is the setup function where the serial port is

nitialized,
  attachInterrupt(0, rpm, RISING); //and the interrupt is attached

// the loop() method runs over and over again,
// as long as the Arduino has power
void loop ()
{
  NbTopsFan = 0;   //Set NbTops to 0 ready for calculations
  sei();      //Enables interrupts
  delay (1000);   //Wait 1 second
  cli();      //Disable interrupts
  Calc = (NbTopsFan * 60 / 5.5); //(Pulse frequency x 60) / 5.5Q, = flow rate

in L/hour
  Serial.print (Calc, DEC); //Prints the number calculated above
  Serial.print (" L/hour\r\n"); //Prints "L/hour" and returns a  new line
}
```

You can refer our forum for more details about Reading Water Flow rate with Water Flow Sensor (http://www.seeedstudio.com/forum/viewtopic.php?f=4&t=989&p=3632#p3632) .

Wiring Diagram

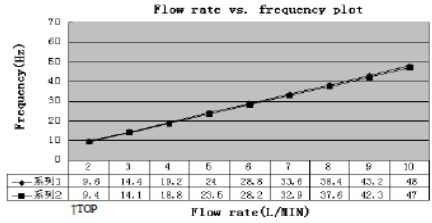
The external diameter of thread the connections use is 1.4mm.



Output Table

Pulse frequency (Hz) in Horizontal Test= 4.8Q, Q is flow rate in L/min. (Results in +/- 3% range)

Output pulse high level	Signal voltage >4.5 V(input DC 5 V)
Output pulse low level	Signal voltage <0.5V(input DC 5V)
Precision	3% (Flow rate from 1L/min to 10L/min)
Output signal duty cycle	40%~60%



FAQ

Here is the Sensors FAQ, people can go here to find questions and answers for this kind of products.

What materials is water flow sensor made of?
Nylon with fiber, avoiding strong acid and strong base.

Is the water flow sensor safe for drinking water?
Yes, it's usage is safe for human consumption. It is frequently used on drinking machines.

Support

If you have questions or other better design ideas, you can go to our forum (http://www.seeedstudio.com/forum) or wish (http://wish.seeedstudio.com) to discuss.

Version Tracker

Revision	Descriptions	Release
v1.0	Initial public release	Feb 14, 2012

Resource

- Reading Water Flow rate with Water Flow Sensor (http://www.seeedstudio.com/forum/viewtopic.php?f=4&t=989&p=3632#p3632)
- Water Flow rate display on LCD (http://www.practicalarduino.com/projects/water-flow-gauge)
- datasheet for the material (http://garden.seeedstudio.com/images/4/4e/YEE70G30HSLNC..pdf)

See Also

LAMPIRAN C
(*Datasheet* Sensor Ultrasonik SRF05)

SRF005 ULTRASONIC RANGE SENSOR

Specification:

The SRF005 ultrasonic range sensor detects objects in it's path and can be used to calculate the range to the object. It is sensitive enough to detect a 3cm diameter broom handle at a distance of over 3m.

Voltage	- 5V
Current	- 30mA Typ. 50mA Max.
Frequency	- 40KHz
Max Range	- 3 m
Min Range	- 3 cm
Sensitivity	- Detect 3cm diameter broom handle at > 3 m
Input Trigger	- 10uS Min. TTL level pulse
Echo Pulse	- Positive TTL level signal, width proportional to range.
Small Size	- 43mm x 20mm x 17mm height

The module can be used in two different modes:

Single Pin	- Single microcontroller pin (08M, and all M2 and X2 parts)
Dual Pin	- Separate PICAXE microcontroller trigger and echo pins

Most users using the latest generation (M2 and X2) PICAXE parts should select 'single pin' connection mode.

Single Pin Connection Mode:

The PICAXE 08M and all M2/X2 parts have bi-directional pins, so the SRF005 can connect to a single I/o pin.

There are two way to achieve this connection on the SRF005, via the 5 way header or via the 3 way header. The 3 way header is designed to be compatible with 'servo extension leads' (e.g. part DAG008) so is often the preferred method on new designs. The 5 way header is compatible with older SRF005 modules/PCBs.

Using the 5 way header (note +5V and 0V are marked on the SRF005):

+5V	Connect to 5V
Not used	Do not connect
Signal	Connect directly to the PICAXE pin
Mode	Connect to 0V
0V	Connect to 0V

Using the 3 way header (note SIG and 0V are marked on the SRF005):

Signal (SIG)	Connect directly to the PICAXE pin
+5V	Connect to 5V
0V	Connect to 0V

When using the 3 pin header you MUST also solder a wire link between the mode and 0V on the 5 way header (ie.a wire link between pads 4 and 5 on the 5 way header).

Take care not to overheat, and therefore damage, the solder connection pads whilst making connections.

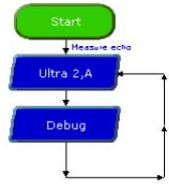
Example PICAXE Program 1:

The following program give an example of how to use the SRF005 module with a PICAXE microcontroller in single pin mode. The special 'ultra' command is designed for use with the SRF005 in single pin mode.

```
symbol SIG = C.1            ; Define pin for Trigger & Echo (All M2, X2 parts)
symbol range = w1           ; 16 bit word variable for range

main:
    ultra SIG,range         ; use dedicated ultra command
    debug range             ; display range via debug command
    pause 50                ; short delay
    goto main               ; loop around forever
```

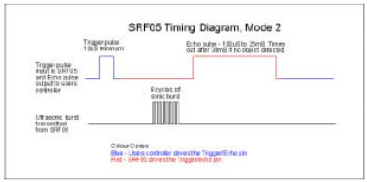
Example Logicator Flowsheet:

The following flowchart give an example of how to use the SRF005 module with a PICAXE microcontroller in single pin mode. The special 'ultra' cell is designed for use with the SRF005.



Technical Details (Single Pin Mode):

The input/Output pin is used to trigger the SRF005 module via a 'pulsout' command and then the pin is converted to an input. The SRF005 module then sends out the sonic burst, and sets the pin high for the time it takes the sonic burst to be returned. Therefore the same PICAXE pin is then used to receive and time this echo pulse via a 'pulsin' command.



The length of the echo pulse is then divided by 5.8 to give a value in cm, and displayed on the computer screen via the 'debug' command. Note that a word variable, w1, is used for the echo timing, as the echo pulse may be a value greater than 255 (maximum value of a byte variable). Word variables are made up of two byte variables and so have a maximum value of 65535 (in this case w1 is made up of b2 and b3, so these two byte variables must not be used anywhere else in the program).

Example Single Pin PICAXE Program 2:

```
symbol SIG = C.1                ; Define pin for Trigger & Echo (All M2, X2 parts)
symbol range = w1               ; 16 bit word variable for range

main:
    pulsout SIG,2               ; produce 20uS trigger pulse (must be minimum of 10uS)
    pulsin SIG,1,range          ; measures the range in 10uS steps
; now convert range to cm (divide by 5.8) or inches (divide by 14.8)
; as picaxe cannot use 5.8, multiply by 10 then divide by 58 instead
    let range = range * 10 / 58 ; multiply by 10 then divide by 58
    debug range                 ; display range via debug command
    pause 50                    ; short delay
    goto main                   ; and around forever

; Note that X2 parts operate at 8MHz instead of 4MHz and so modify the calculation
; let range = range * 5 / 58  ; multiply by (10/2 = 5) then divide by 58
```

Dual Pin Mode - separate trigger / echo microcontroller pins:

The dual pin mode is used for older PICAXE chips such as the 18X or 28X1.
The SRF005 ultrasonic range finder has 5 connections pins. The 3 pin connector is not used in dual pin mode.

Using the 5 way header (note +5V and 0V are marked on the SRF005):

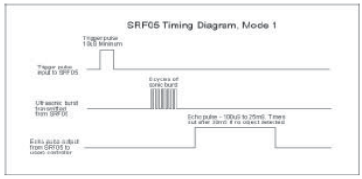
+5V	Connect to 5V
Echo	Connect directly to PICAXE input pin
Trigger	Connect directly to PICAXE output pin
Mode	Do not connect
0V	Connect to 0V

Important - Note that the 'Mode' (pin 4) connection **MUST NOT** be connected for correct operation in this separate trigger/echo mode.

Take care not to overheat, and therefore damage, the solder connection pads whilst making connections.

The SRF005 **Echo Output** is connected to a PICAXE **input** pin.
The SRF005 **Trigger Input** is connected to a PICAXE **output** pin. Note this must be a direct connection to the PICAXE chip leg (do not connect via a darlington driver buffered output on a PICAXE project board).

The following program gives an example of how to use the SRF005 module with a PICAXE microcontroller. Output 3 is used to trigger the SRF005 module via a 'pulsout' command. The SRF005 module then sends out the sonic burst, and sets the Echo Output connection high for the time it takes the sonic burst to be returned. Therefore the PICAXE input (input 6) is used to receive and time this echo pulse via a 'pulsin' command.



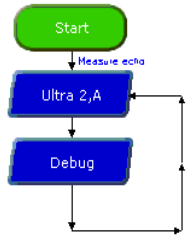
The length of the echo pulse is then divided by 5.8 to give a value in cm, and displayed on the computer screen via the 'debug' command. Note that a word variable, w1, is used for the echo timing, as the echo pulse may be a value greater than 255 (maximum value of a byte variable). Word variables are made up of two byte variables and so have a maximum value of 65535 (in this case w1 is made up of b2 and b3, so these two byte variables must not be used anywhere else in the program).

Sample Dual Pin Mode PICAXE Program:

```
symbol trig = 3          ; Define output pin for Trigger pulse (A, M, X, X1 parts)
; symbol trig = b.3      ; Define output pin for Trigger pulse (M2, X2 parts)
symbol echo  = 6         ; Define input pin for Echo pulse (A, M, X, X1 parts)
; symbol echo  = c.6     ; Define input pin for Echo pulse (M2, X2 parts)
symbol range = w1        ; 16 bit word variable for range

main:
    pulsout trig,2       ; produce 20uS trigger pulse (must be minimum of 10uS)
    pulsin echo,1,range  ; measures the range in 10uS steps
    pause 20             ; recharge period after ranging completes
; now convert range to cm (divide by 5.8) or inches (divide by 14.8)
; as picaxe cannot use 5.8, multiply by 10 then divide by 58 instead
    let range = range * 10 / 58      ; multiply by 10 then divide by 58
    debug range                      ; display range via debug command
    goto main                        ; and around forever

; Note that X2 parts operate at 8MHz instead of 4MHz and so modify the calculation
; let range = range * 5 / 58  ; multiply by (10/2 = 5) then divide by 58
```

Example Logicator Flowsheet:

The following flowchart give an example of how to use the SRF005 module with a PICAXE microcontroller in dual pin mode. The special 'ultra' cell is designed for use with the SRF005 and will automatically enable dual pin mode for those PICAXE chips that require it.



LAMPIRAN D
(Pemrograman pada CodeVision)

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.3 Standard
Automatic Program Generator
© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 07/07/2018
Author  : tyery08
Company : embeeminded.blogspot.com
Comments:


Chip type               : ATmega128A
Program type            : Application
AVR Core Clock frequency: 16,000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 1024
*****************************************************/

#include <mega128a.h>
#include <alcd.h>
#include <stdlib.h>
#include <delay.h>
#include <stdio.h>
#include <i2c.h>
#include <ds1307.h>

#define pompa1  PORTD.4
#define pompa2  PORTD.5
#define mov1    PORTB.6
#define mov2    PORTB.7
#define b3 PORTA.0
#define b2 PORTA.2
#define b1 PORTA.4
#define b0 PORTA.6

char temp1 [16];
```

```
int waktu = 0,persen;
unsigned char Rpm, Rpm2;
float jrk,acuan_mov; //,acuan_mov_min;
int jarak,count=0,drajat=0,x,y;
char buff[32], buff3[50], buff4[50];
float freq,volt,adc,adc_arus;  // to store value of frequency value
unsigned char i=0,dur;
int tegangan;
float av_adc,av_adc1,vout,arus,daya;
char buffer[32], buffer1[10]; // to store the frequency value as a string to be displayed on
lcd
int counter=0,flag=0;
int flow;
float flow_rate;
int max_step=150;
int hitungstep;
char temp[16], temp1[16];
unsigned char wd;
unsigned char dd;
unsigned char mm;
unsigned char yy;
unsigned char s;
unsigned char m;
unsigned char h;

// External Interrupt 4 service routine
interrupt [EXT_INT4] void ext_int4_isr(void)
{
// Place your code here
   counter++;
//counter1++;
}

#ifndef RXB8
#define RXB8 1
#endif

#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif
```

```
#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
#endif

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// Get a character from the USART1 Receiver
#pragma used+
char getchar1(void)
{
char status,data;
while (1)
      {
      while (((status=UCSR1A) & RX_COMPLETE)==0);
      data=UDR1;
      if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
        return data;
      }
}
#pragma used-

// Write a character to the USART1 Transmitter
#pragma used+
void putchar1(char c)
{
while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
UDR1=c;
}
#pragma used-
```

```
#define_ALTERNATE_PUTCHAR_
#include <stdio.h>
#define USART0 0              // agar pembacaan tidak acak
#define USART1 1
unsigned char poutput;

void putchar(char c)
{
switch (poutput)
{
case USART0: // the output will be directed to USART0
while ((UCSR0A & DATA_REGISTER_EMPTY)==0);
UDR0=c;
break;

case USART1: // the output will be directed to USART1
while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
UDR1=c;
break;
};
}

// Standard Input/Output functions
#include <stdio.h>

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Reinitialize Timer 0 value
TCNT0=0x8A;
// Place your code here
   waktu++;
   if (waktu >=120){//kira2 bisa mencapai 1 detik
//detik++;
Rpm = counter * 60.0;
Rpm2 = Rpm * 4.0;
counter = 0;
waktu = 0;
Rpm2 = Rpm2;
}
}

// Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
```

```
{
// Place your code here
  i++;
}

#define ADC_VREF_TYPE 0x60

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

// Declare your global variables here
void baca_volt(){
 for(i=0;i<10;i++){
   adc=read_adc(0);
   av_adc=av_adc+adc;
 }
 av_adc=av_adc/10;
 volt=av_adc*106/255;

}
void baca_arus (){
 for (i=0;i<10;i++){
 adc_arus=read_adc(1);
 av_adc1=av_adc1+adc_arus;
   }
   av_adc1=av_adc1/10;
   vout=av_adc1*106/255;
   arus=fabs(vout-2.50)/0.066/1000;
   daya=arus*volt;
   }
void baca_ultrasonic(){
   count=0;
   PORTA.1=1;
```

```
    delay_us(15);
    PORTA.1=0;
    while(PINA.3==0){};
    while(PINA.3==1){count++;};
    jrk=count;
    jrk=jrk/100;
    jrk=66.5-jrk;
    jarak = jrk;
    /*if (jrk<0){
      lcd_gotoxy(0,0);
      lcd_putsf("ERROR");}
    else if (jrk>70)
      { lcd_gotoxy(0,0);
       lcd_putsf("ERROR");}
    else {
    lcd_gotoxy(0,0);
    lcd_putsf("Lvl:");
    //sprintf(buffer,"%d ",jrk);
    ftoa(jrk,2,buffer);
    lcd_puts(buffer);

    lcd_gotoxy(10,0);
    lcd_putsf("Lvl:");
    sprintf(buffer,"%d ",jarak);
    lcd_puts(buffer);
    lcd_putsf(" Cm ");
    delay_ms(100);}*/
  }
  void baca_flow(){
   TIMSK=0x05;
   TCCR1B=0x07;
   delay_ms(100);
   TCCR1B=0x00;
   TIMSK=0x00;
   dur=TCNT1;
   //flow_rate=0;
   freq = (((dur + i*65536)*600)/4.8)*0.0166;
   TCNT1=0x0000;
   i=0;
   flow_rate=freq;
   flow_rate=flow_rate-1359872.00;
   flow = flow_rate;
   lcd_gotoxy(0,2);
   lcd_putsf("F:");
   //sprintf(buffer,"%d",flow_rate-18350);
```

```
  ftoa(flow_rate,2,buffer);
  lcd_puts(buffer);
  lcd_putsf("L/M  ");
 }
 void cecewe_start(){
    b0=0; b1=0; b2=0; b3=1;
  delay_ms(5);
  b0=0; b1=0; b2=1; b3=0;
  delay_ms(5);
  b0=0; b1=1; b2=0; b3=0;
  delay_ms(5);
  b0=1; b1=0; b2=0; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=0; b3=1;
  delay_ms(5);
  b0=0; b1=0; b2=1; b3=0;
  delay_ms(5);
  b0=0; b1=1; b2=0; b3=0;
  delay_ms(5);
  b0=1; b1=0; b2=0; b3=0;
  delay_ms(5);
 }
 void cecewe(){  //close
  b0=0; b1=0; b2=0; b3=1;
  delay_ms(5);
  b0=0; b1=0; b2=1; b3=0;
  delay_ms(5);
  b0=0; b1=1; b2=0; b3=0;
  delay_ms(5);
  b0=1; b1=0; b2=0; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=0; b3=1;
  delay_ms(5);
  b0=0; b1=0; b2=1; b3=0;
  delay_ms(5);
  b0=0; b1=1; b2=0; b3=0;
  delay_ms(5);
  b0=1; b1=0; b2=0; b3=0;
  delay_ms(5);
  hitungstep--;
  persen=(hitungstep*100)/max_step;
  if(persen<0){
    persen=0;
  }
 /* lcd_gotoxy(13,1);
```

```
  sprintf(buffer,"%d  ",persen);
  lcd_puts(buffer);  */

}

void cewe(){    //open
  b0=1; b1=0; b2=0; b3=0;
  delay_ms(5);
  b0=0; b1=1; b2=0; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=1; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=0; b3=1;
  delay_ms(5);
  b0=1; b1=0; b2=0; b3=0;
  delay_ms(5);
  b0=0; b1=1; b2=0; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=1; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=0; b3=1;
  delay_ms(5);
  hitungstep++;
  persen=(hitungstep*100)/max_step;
  if(persen>100){
    persen=100;
  }
 /* lcd_gotoxy(13,1);
  sprintf(buffer,"%d  ",persen);
  lcd_puts(buffer);*/
}
void stopped(){
  b3=0; b2=0; b1=0; b0=0;
}
void tampil_persen(){

}

void mov(int kondisi){
    if(kondisi == 1){ //MOV CLOSE
      mov1 = 1;
      mov2  = 0;
        drajat -= 50;
        delay_ms(300);
    }
```

```
    if(kondisi == 2){ //MOV OPEN
      mov1 = 0;
      mov2  = 1;
        drajat += 50;
        delay_ms(2400);
    }
    if(kondisi == 3){ //MOV no operation
      mov1 = 1;
      mov2  = 1;
    }     }

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA.1=0;
PORTA.3=1;
DDRA.0=1;
DDRA.1=1;
DDRA.2=1;
DDRA.3=0;
DDRA.4=1;
DDRA.6=1;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0xC0;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD.4=1;
```

```
DDRD.5=1;

// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTE=0x00;
DDRE=0x00;

// Port F initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTF=0x00;
DDRF=0x00;

// Port G initialization
// Func4=In Func3=In Func2=In Func1=In Func0=In
// State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 250,000 kHz
// Mode: Normal top=0xFF
// OC0 output: Disconnected
ASSR=0x00;
TCCR0=0x07;
TCNT0=0x8A;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 16000,000 kHz
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: On
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
```

```
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer3 Stopped
// Mode: Normal top=0xFFFF
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
```

```
OCR3CH=0x00;
OCR3CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: On
// INT4 Mode: Rising Edge
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x03;
EIMSK=0x10;
EIFR=0x10;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x05;

ETIMSK=0x00;

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 9600
UCSR0A=0x00;
UCSR0B=0x98;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;

// USART1 initialization
UCSR1A=0x00;
UCSR1B=0x18;
UCSR1C=0x06;
UBRR1H=0x00;
UBRR1L=0x67;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
```

```
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 1000,000 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

i2c_init();
rtc_init(0,0,0);

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTC Bit 0
// RD - PORTC Bit 1
// EN - PORTC Bit 2
// D4 - PORTC Bit 3
// D5 - PORTC Bit 4
// D6 - PORTC Bit 5
// D7 - PORTC Bit 6
// Characters/line: 20
lcd_init(20);

// Global enable interrupts
#asm("sei")

/*****setup time and date*****/
h=22;m=28;s=00;
rtc_set_time(h,m,s);

dd=16;mm=07;yy=18;
rtc_set_date(wd,dd,mm,yy);
/*****************************/
```

```
pompa1=1;
pompa2=0;
mov1=0;
mov2=1;
lcd_gotoxy(0,0);
lcd_putsf("     WELCOME TO      ");
lcd_gotoxy(0,1);
lcd_putsf("     PLTMH PLANT     ");
lcd_gotoxy(0,2);
lcd_putsf("        2018         ");
for(i=0;i<160;i++){
   cecewe_start();
}

delay_ms(1500);
lcd_clear();
while (1)
      {
      // Place your code here
      start:
      baca_ultrasonic();
      baca_flow();
      baca_volt();
      baca_arus();

      lcd_gotoxy(0,0);
      lcd_putsf("L:");
      sprintf(buff,"L:%d cm ",jarak);
      ftoa(jrk,2,buffer);
      lcd_puts(buffer);
      lcd_putsf("cm");

      lcd_gotoxy(11,0);
      lcd_putsf("V:");
      ftoa(volt,2,buff);
      lcd_puts(buff);
      lcd_putsf("V");

      sprintf(temp1,"N:%d RPM", Rpm);
      lcd_gotoxy(0,1);
      lcd_puts(temp1);

      lcd_gotoxy(11,1);
      lcd_putsf("I:");
```

```
      ftoa(arus,2,buff);
      lcd_puts(buff);
      lcd_putsf("A");

      lcd_gotoxy(11,2);
      lcd_putsf("P:");
      ftoa(daya,2,buff);
      lcd_puts(buff);
      lcd_putsf("W");

      lcd_gotoxy(11,3);
      sprintf(buffer,"MOV2:%d",persen);
      lcd_puts(buffer);
      lcd_putsf("%");

      lcd_gotoxy(0,3);
      sprintf(buffer1,"N2:%d Rpm",Rpm2);
      lcd_puts(buffer1);
      lcd_putsf("%");

      //=====data logger======
      sprintf(buff3, "%02u:%02u:%02u/%02u:%02u:%02u ; %.2f cm ; %d RPM ; %.2f lpm ;
%.2f V\r",dd,mm,yy,s,m,h, jrk, Rpm, flow_rate, volt);
      poutput = USART1;
      puts(buff3);


//================================================HMI==========
==============================
     // matikan salah satu kodingan, karena dalam kodingan ini kedua HMI belum
diintegrasikan

      /*//======HMI delima lama=====
      sprintf(buff4, "%.2f;%d\r", volt, Rpm);
      poutput = USART0;
      puts(buff4);*/

      // hmi baru
      poutput = USART0;
      printf("%.2f;%d;%d;%.2f;%.2f\r",jrk,Rpm,Rpm2,flow_rate,volt);
      puts(buff4);

      /*======HMI ragil lama=====
      //sprintf(buff4, "JARAK: %.2f; RPM: %d; FLOW: %.2f; TEGANGAN: %.2f\r", jrk,
Rpm, flow_rate, volt);
```

```
    //poutput = USART0;
    //puts(buff4);*/

//================================================================
================================
    // delay_ms(300);

     if(volt<38){
            cewe();
            tampil_persen();
          }
          else if(volt>38.5){
            cecewe();
            tampil_persen();
          }
          else {
            stopped();
            tampil_persen();
          }
      acuan_mov= jrk - 35;


           if(jarak==35 && flag==0){
         pompa1 =0;  //pompa1
         pompa2 =0;  //pompa2
          mov1=0;
          delay_ms(1500);
          mov1=1;
          flag=1;
          delay_ms(1000);
          goto start;
       }

   if(jarak>45 && flag==1){
     pompa1=1;
     pompa2=1;

   }
   else if(jarak>35 && jarak<45 && flag==1){
     pompa1=0;
     pompa2=0;
     mov2=0; //
     mov1=1;
     drajat=0;
     x=1;
```

```
    y=0;
   }
   else if(jarak==35 && flag==1 && x==1){
    pompa1=0;
    pompa2=0;
    if(drajat < 50){
        mov(2);     //MOV OPEN
        mov(3);
     }
    }
   else if(jarak==35 && flag==1&& y==1){
    pompa1=0;
    pompa2=0;
    if(drajat > 50){
        mov(1);     //MOV OPEN
        mov(3);

     }
   }
   else if(jarak>25 && jarak<35 && flag==1){
    pompa1=0;
    pompa2=0;
    mov1=1; //
    mov2=0;
    drajat=100;
    x=1;
    y=0;
   }
   else if(jarak<25 && flag==1){
    pompa1=1;
    pompa2=1;

   } //delay_ms(250);
   }
}
```

LAMPIRAN E
(Pemrograman HMI pada Visual Studio)

```
Imports System.IO
Imports System.IO.Ports
Imports System.Threading
Imports ZedGraph

Public Class Form1
    Dim portOpen As Boolean
    Dim dataMasuk As String
    Dim data0, data1, data2, data3, data4
    Dim nilai_tegangan, nilai_kecepatan As Integer
    Public Penerima

    'Variable zedgraph
    Dim graphane1 As GraphPane
    Dim list1, list2, list3, list4, list5 As RollingPointPairList
    Dim garisNilai As LineItem
    Dim starting_time As Double = 100

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim myPort As Array

        myPort = IO.Ports.SerialPort.GetPortNames()
        Timer2.Enabled = False

        ComboBox2.Items.Add(9600)
        ComboBox2.Items.Add(19200)
        ComboBox2.Items.Add(38400)
        ComboBox2.Items.Add(57600)
        ComboBox2.Items.Add(115200)
        For i = 0 To UBound(myPort)
            ComboBox1.Items.Add(myPort(i))
        Next
        btn_disconnect.Enabled = False

        'zedgraph
        starting_time = Environment.TickCount
        graphane1 = ZedGraphControl1.GraphPane
        graphane1.Title.Text = "Grafik Pembacaan Sensor"
        graphane1.XAxis.Title.Text = "Timing (Second)"
        graphane1.YAxis.Title.Text = "CM / RPM / RPM / LPS / VOLT"
```

```
        list1 = New RollingPointPairList(120)
        list2 = New RollingPointPairList(120)
        list3 = New RollingPointPairList(120)
        list4 = New RollingPointPairList(120)
        list5 = New RollingPointPairList(120)

        garisNilai = graphane1.AddCurve("Jarak", list1, Color.Red,
SymbolType.None)
        garisNilai = graphane1.AddCurve("RPM Turbin", list2, Color.Blue,
SymbolType.None)
        garisNilai = graphane1.AddCurve("RPM Generator", list3, Color.Orange,
SymbolType.None)
        garisNilai = graphane1.AddCurve("Flow", list4, Color.Aqua,
SymbolType.None)
        garisNilai = graphane1.AddCurve("Tegangan", list5, Color.Pink,
SymbolType.None)

        btn_play.Enabled = False
        btn_paus.Enabled = False

    End Sub

    Private Sub SerialPort1_DataReceived(sender As Object, e As
SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived

        Try
            dataMasuk = SerialPort1.ReadExisting()
            'dataMasuk = SerialPort1.ReadLine()

        Catch ex As Exception

        End Try

    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles
Timer1.Tick

        lbl_data.Text = dataMasuk

        Try
            Dim nilai() As String = Data(dataMasuk)
            data0 = nilai(0)
            data1 = nilai(1)
            data2 = nilai(2)
```

```
            data3 = nilai(3)
            data4 = nilai(4)

            tb_rpm.Text = data1 + " Rpm"
            tb_flow.Text = data3 + " Lps"
            tb_level.Text = data0 + " Cm"
            tb_tegangan.Text = data4 + " Volt"
        Catch ex As Exception

        End Try

    End Sub

    Private Sub Timer2_Tick(sender As Object, e As EventArgs) Handles
Timer2.Tick

        Dim jarak, RPM, RPM1, flow, tegangan As Single

        jarak = data0
        RPM = data1
        RPM1 = data2
        flow = data3
        tegangan = data4

        Dim xscale As Scale
        Dim kurvaJarak As LineItem = ZedGraphControl1.GraphPane.CurveList(0)
        Dim kurvaRPM As LineItem = ZedGraphControl1.GraphPane.CurveList(1)
        Dim kurvaRPM1 As LineItem =
ZedGraphControl1.GraphPane.CurveList(2)
        Dim kurvaFlow As LineItem = ZedGraphControl1.GraphPane.CurveList(3)
        Dim kurvaTegangan As LineItem =
ZedGraphControl1.GraphPane.CurveList(4)
        Dim list1 As IPointListEdit = kurvaJarak.Points
        Dim list2 As IPointListEdit = kurvaRPM.Points
        Dim list3 As IPointListEdit = kurvaRPM1.Points
        Dim list4 As IPointListEdit = kurvaFlow.Points
        Dim list5 As IPointListEdit = kurvaTegangan.Points
        Dim waktu As Double = (Environment.TickCount - starting_time) / 1000.0

        If CheckJarak.Checked = True Then
            list1.Add(waktu, jarak)
        End If

        If checkRPM.Checked = True Then
            list2.Add(waktu, RPM)
        End If
```

```
        If CheckRPM1.Checked = True Then
            list3.Add(waktu, RPM1)
        End If

        If CheckFlow.Checked = True Then
            list4.Add(waktu, flow)
        End If

        If checktegangan.Checked = True Then
            list5.Add(waktu, tegangan)
        End If

        xscale = ZedGraphControl1.GraphPane.XAxis.Scale
        If (waktu > xscale.Max - xscale.MajorStep) Then
            xscale.Max = waktu + xscale.MajorStep
            xscale.Min = xscale.Max - 10
        End If

        ' Pastikan Y axis di scale ulang untuk mengakomodir data aktual
        ZedGraphControl1.AxisChange()

        ' Redraw paksa
        ZedGraphControl1.Invalidate()


    End Sub

    Private Sub btn_connect_Click(sender As Object, e As EventArgs) Handles
btn_connect.Click

        If ComboBox1.Text = "" Then
            MsgBox("CONNECT THE MINSIS BROO...")

        Else : SerialPort1.PortName = ComboBox1.Text
            SerialPort1.BaudRate = ComboBox2.Text
            SerialPort1.Open()
            SerialPort1.Encoding = System.Text.Encoding.Default
            Timer1.Enabled = True
            btn_disconnect.Enabled = True
            btn_connect.Enabled = False
            btn_play.Enabled = True
            lbl_status.Text = "Connected"
            ComboBox1.Enabled = False
            ComboBox2.Enabled = False
        End If
```

```
    End Sub

    Private Sub btn_disconnect_Click(sender As Object, e As EventArgs) Handles
btn_disconnect.Click

        SerialPort1.Close()
        Timer1.Enabled = False
        portOpen = False
        lbl_status.Text = "Disconnected"
        ComboBox1.Enabled = True
        ComboBox2.Enabled = True
        btn_connect.Enabled = True
        btn_disconnect.Enabled = False

    End Sub

    Function Data(ByVal Penerima As String) As String()

        Dim Pembatas As String = ";"c
        Dim data_disaring() As String = Penerima.Split(Pembatas)
        Return {data_disaring(0), data_disaring(1), data_disaring(2),
data_disaring(3), data_disaring(4)}

    End Function

    Private Sub btn_play_Click(sender As Object, e As EventArgs) Handles
btn_play.Click

        Timer2.Enabled = True
        btn_play.Enabled = False
        btn_paus.Enabled = True

    End Sub

    Private Sub btn_paus_Click(sender As Object, e As EventArgs) Handles
btn_paus.Click

        Timer2.Enabled = False
        btn_play.Enabled = True
        btn_paus.Enabled = False

    End Sub

    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles
Button3.Click
```

```
        ZedGraphControl1.GraphPane.CurveList(0).Clear()
        ZedGraphControl1.GraphPane.CurveList(1).Clear()
        ZedGraphControl1.GraphPane.CurveList(2).Clear()
        ZedGraphControl1.GraphPane.CurveList(3).Clear()
        ZedGraphControl1.GraphPane.CurveList(4).Clear()
        checktegangan.Checked = False
        checkRPM.Checked = False

    End Sub

    Private Sub bt_connect_Click(sender As Object, e As EventArgs) Handles
bt_connect.Click
        TabControl1.SelectTab(1)
    End Sub

    Private Sub bt_FlowLevel_Click(sender As Object, e As EventArgs) Handles
bt_FlowLevel.Click
        TabControl1.SelectTab(3)
    End Sub

    Private Sub bt_RPMTegangan_Click(sender As Object, e As EventArgs)
Handles bt_RPMTegangan.Click
        TabControl1.SelectTab(4)
    End Sub

    Private Sub bt_grafik_Click(sender As Object, e As EventArgs) Handles
bt_grafik.Click
        TabControl1.SelectTab(2)
    End Sub

    Private Sub bt_home_Click(sender As Object, e As EventArgs) Handles
bt_home.Click
        TabControl1.SelectTab(0)
    End Sub

    Private Sub Button4_Click(sender As Object, e As EventArgs) Handles
Button4.Click
        TabControl1.SelectTab(0)
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
        TabControl1.SelectTab(0)
    End Sub
```

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        TabControl1.SelectTab(0)
    End Sub
End Class
```

LAMPIRAN F
(Karakteristik Statik Pembacaan Sensor *Water Flow*)

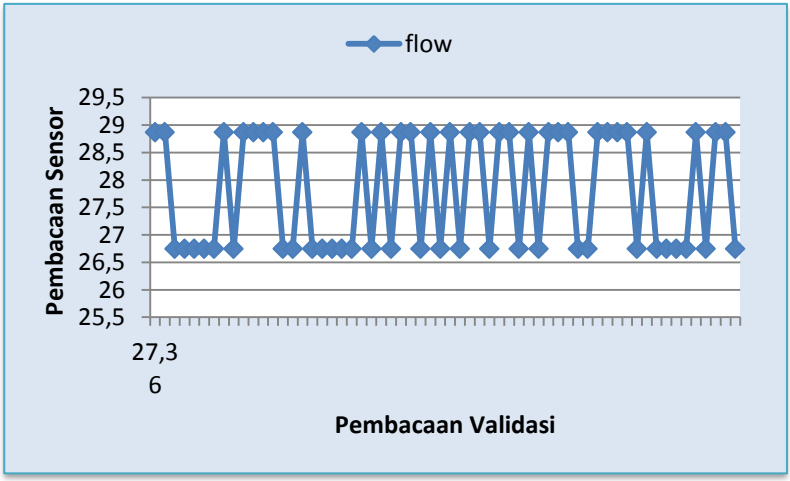
Untuk mencari karakteristik statik pembacaan *flow*, dilakukan validasi dengan validator berupa perhitungan debit. Pengambilan data dilakukan selama 1 menit. Dalam 1 menit tersebut, hasil pembacaan sensor akan di rata – rata dan dibandingkan dengan pembacaan perhitungan. Berikut perhitungan debit selama 1 menit.

Q = v / t
= 27,36 / 60
= 0,456 liter / detik
= 27,36 liter / menit

V diperoleh dari volume tangki, yaitu:
V = p x l x t (awal – akhir)
= 60 x 60 x (43,3 – 35,7)
= 27360 cm^3
= 27,36 liter

No.	Pembacaan Perhitungan (L/menit)	Pembacaan Sensor (L/menit)	No.	Pembacaan Perhitungan (L/menit)	Pembacaan Sensor (L/menit)
1.	27,36	28,87	31.	27,36	28,87
2.		28,87	32.		26,75
3.		26,75	33.		28,87
4.		26,75	34.		26,75
5.		26,75	35.		28,87
6.		26,75	36.		28,87
7.		26,75	37.		26,75
8.		26,75	38.		28,87
9.		26,75	39.		28,87
10.		28,87	40.		26,75
11.		26,75	41.		28,87

12.		28,87	42.		26,75
13.		28,87	43.		28,87
14.		28,87	44.		28,87
15.		28,87	45.		28,87
16.		26,75	46.		26,75
17.		26,75	47.		26,75
18.		28,87	48.		28,87
19.		26,75	49.		28,87
20.		26,75	50.		28,87
21.		28,87	51.		28,87
22.		26,75	52.		26,75
23.		26,75	53.		28,87
24.		28,87	54.		26,75
25.		26,75	55.		26,75
26.		28,87	56.		26,75
27.		26,75	57.		26,75
28.		28,87	58.		28,87
29.		28,87	59.		26,75
30.		26,75	60.		28,87
Rata - rata					27,81



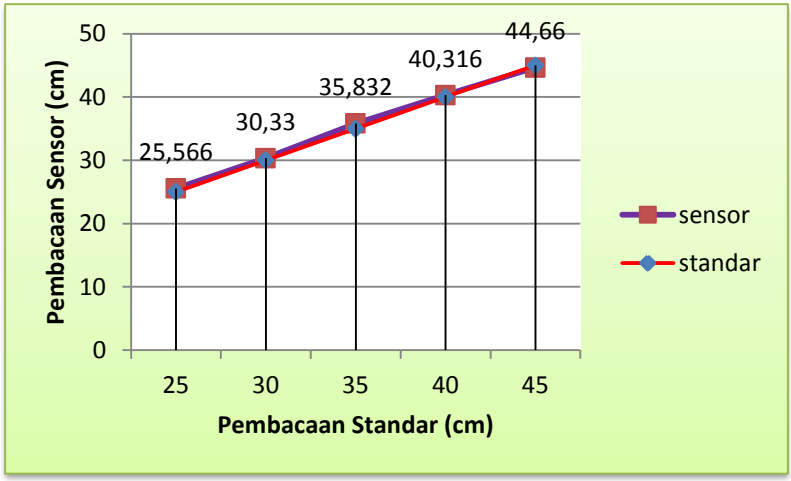
Dari data tersebut dapat diketahui karakteristik statik dengan menggunakan persamaan yang ada pada sub bab 2.7 sebagai berikut.

No.	Karakteristik Statik	Hasil
1.	Range	26,75 L/menit – 28,87 L/menit
2.	Span	2,12 L/menit
3.	Sensitivitas	0.077485
4.	Linieritas	25,23585 %
5.	Error	1,645 %
6.	Akurasi	98,355 %

LAMPIRAN G
(Karakteristik Statik Pembacaan Sensor Ultrasonik)

Untuk mencari karakteristik statik pembacaan *level*, dilakukan validasi dengan validator berupa penggaris. Pembacaan dilakukan dengan 5 variasi ketinggian yang berbeda.

Nomor	Pembacaan Sensor *Level* (cm)	Variasi Level (cm)
1.	44,66	45
2.	40,316	40
3.	35,832	35
4.	30,33	30
5.	25,566	25



Dari data tersebut dapat diketahui karakteristik statik dengan menggunakan persamaan yang ada pada sub bab 2.7 sebagai berikut.

No.	Karakteristik Statik	Hasil
1.	Range	25 cm – 45 cm
2.	Span	20 cm
3.	Sensitivitas	0,9547
4.	Linieritas	17,79 %
5.	Error	0,974 %
6.	Akurasi	99,026 %

BIODATA PENULIS

Penulis bernama Ragil Suryaning Fajar. Lahir di Sidoarjo pada tangal 06 September 1997, merupakan anak terakhir dari tiga bersudara. Penulis telah menempuh pendidikan formal di SDN Pilang 1, SMPN 1 Wonoayu, dan MAN Sidoarjo. Kemudian penulis mengikuti seleksi masuk D3 ITS pada tahun 2015 dan diterima di Program Studi D3 Metrologi dan Instrumentasi, Jurusan Teknik Fisika, Fakultas Teknologi Industri yang sekarang telah berubah menjadi Departemen Teknik Instrumentasi, Fakultas Vokasi.

Selama menempuh perkuliahan, penulis aktif mengikuti kegiatan kemahasiswaan. Mulai dari staff PSDM HMTF 16/17, staff PSDM BEM ITS 16-18, ketua divisi PSDM Himatekins 17/18, HRD Gerigi ITS 2017, wakil ketua eksternal BEM FV 2018, dan LKMM sampai Tingkat Menengah (TM). Serta beberapa event mulai dari jurusan, fakultas, hingga institut. Penulis juga pernah melaksanakan kerja praktek selama 2 bulan di PT Sentra Elektra Marindo yang berlokasi di Perak, Surabaya. Apabila terdapat kritik dan saran, penulis dapat dihubungi melalui email ragilsuryaningfajar@gmail.com.